UNIVERSIDADE DO MINDELO
DEPARTAMENTO de ENGENHARIA E RECURSOS DO MAR

CURSO DE LICENCIATURA EM INFORMÁTICA DE GESTÃO

RELATÓRIO DE PROJETO DE LICENCIATURA

Ano letivo 2015/2016 – 4º Ano

Autor: **Emerson Jorge Lopes do Rosário Sequeira, Nº 2127.**

Mindelo, 2016

Monografia apresentada a Universidade do Mindelo como parte dos requisitos para obtenção do grau de licenciatura no Curso de Licenciatura em Informática de Gestão.

São Vicente, Mindelo Junho de 2016

Emerson Jorge Lopes do Rosário Sequeira

Orientador: Ciríaco Brito

Agradecimentos

Numa caminhada seja ela qual for, muito dificilmente será feita apenas pelo caminhante.

Para alguns menos humildes, dizer que "cheguei até aqui sozinho" é esquecer os suportes emocionais, financeiros e intelectual que nos rodeiam e todo o trabalho elaborado, na definição de um profissional.

Os agradecimentos passam pelos que fortalecem os conteúdos, criando uma base solida com frutos futuros na construção de uma sociedade democrata e inovadora.

A estas palavras dedico:

---------- Arlinda do Rosário

---------- Elton Sequeira

---------- Ciríaco Brito

---------- Docente João Firmino

---------- Docente Samuel Lima

---------- Docente Luís Graça

---------- José António Lima Barber

"Se eu vi mais longe foi por estar sobre ombros de gigantes"

- Isaac Newton -

I. Resumo

Aplicação Web pode-se dizer que é uma aplicação que responde a pedidos do utilizador através de um Browser. As aplicações web orientados a objetos hoje em dia têm cada vez mais espaços nas empresas. A redução dos custos de operacionalidade e a gestão das informações, depende da visão e missão das empresas n desenvolvimento das aplicações *web*, enquanto uma aplicação desktop precisaria de outros meios para a sua manutenção e com maiores custos. A facilidade do acesso principalmente na ASA em que os nossos servidores estão espalhados pelos aeroportos internacionais em Cabo Verde, sem a necessidade de ser instalada num computador sendo o acesso feito apenas através do *Browser*. Há uma grande vantagem na atualização que só deve ser feita no servidor em vez de máquina a maquina, também como a sua escalabilidade.

A aplicação escolhida para desenvolvimento vai ajudar na gestão da frota das viaturas do corpo de Bombeiros do Aeroporto Internacional Cesária Évora, designada de ***GestFrota***, com programação em PHP, HTML, CSS, JQuery e SQL.

O trabalho permite o preenchimento diário de check-list dos equipamentos, o registo das viaturas e a gestão de extintores entre outros.

Palavras-chaves – Sistema de Gestão de Quarteis de Bombeiros e Frotas, informações, base de Dados, engenharia de *software*.

II. Abstract

Web application can be said that is an application that responds to user requests through a browser. Web applications object-oriented nowadays have increasingly spaces in companies. The reduction of operational costs and information management to depend on the vision and mission of companies for development of web applications, while a desktop application need other means for their maintenance and higher costs. The ease of access especially in ASA that our servers are spread by international airports in Cape Verde, without the need to be installed on a computer with access made only through the browser. There is a great advantage in the update that only should be done on the server instead of machine-to-machine, as well as its scalability.

The application chosen for development will help in the management of the fleet of vehicles of the Aircraft and Rescue Firefighter Squad of the International Airport Fire Cesária Évora, designated GestFrota programmed in PHP, HTML, CSS, JQuery and SQL.

The work allows the daily fill checklist of equipment, registration of vehicles and extinguishers management among others.

Keywords – Firefighter Quarter Management System, information, database, software engineering.

Índice

INDICE DE FIGURAS

Índice de Tabelas

CAPITULO 1

1 INTRODUÇÃO

Nos dias de hoje as aplicações Web têm um papel importante nas empresas. A satisfação do cliente passou a ser uma grande prioridade, um mercado de concorrência agressiva e cada vez mais apertado. Vê-se também preocupado com a satisfação do cliente interno cada vez mais exigente e competitivo, a resposta de melhoria continua e crescimento económico.

No mercado atual as empresas têm que saber vender os seus serviços, para isso é preciso investir na resposta e solicitação para a satisfação do cliente para que os objetivos sejam alcançados e os benefícios sejam benéficas as empresas.

O uso das aplicações Web veio revolucionar e melhorar os suportes de gestão, quebrando a dependência dos colaboradores dos computadores dos serviços, uma vez que podem ter acesso ao sistema em qualquer parte desde que haja internet.

1.1 Objetivo Geral

O objetivo geral do trabalho é o desenvolvimento de uma aplicação para a gestão do Quartel dos Bombeiros do Aeroporto Internacional Cesária Évora.

Visão Geral do Sistema

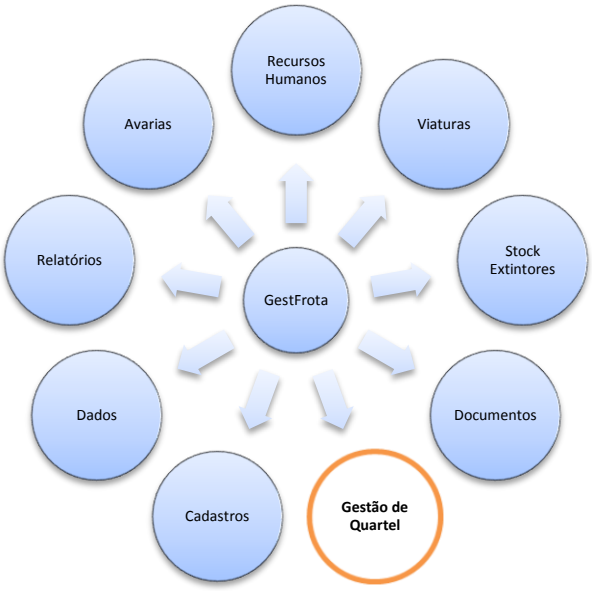


Figura 1 – Fluxo da Visão Geral do Sistema

1.2 Objetivos Específicos

- Gerir a frota das viaturas existente no serviço dos Bombeiros;
- Gerir, cadastrar e organizar os extintores do Aeroporto;
- Criar o cadastro dos Bombeiros para que as informações estejam a mão num único canal (o aplicativo);
- Criar e organizar o cadastro de viaturas;
- Registar os reportes que possam aparecer no serviço;
- Registar e gerir as ocorrências através de formulários de relatórios;
- Organizar os dados referentes a gastos no quartel;
- Ter um arquivo de dados do consumo de combustível por viatura.

1.3 O sistema deve:

Permitir que todos os dados do quartel estejam num único aplicativo ou canal de informação.

Deve permitir que a gestão as ocorrências seja feita de forma organizada seja ela através dos códigos das viaturas ou através do supervisor de serviço em que os dados são encontrados através de filtros.

Melhorar a qualidade do serviço e tempo de resposta no tratamento de informações.

Sendo que ainda não existe nenhuma aplicação de suporte de gestão nos quarteis de Bombeiros nos aeroportos de Cabo Verde, a necessidade atual, mostra que a dificuldade no tratamento de informações aumenta dia a dia tendo em conta o aumento da carga de trabalho que os aeroportos são submetidos.

1.4 Motivação

Pelo fato de não haver uma aplicação na gestão das informações nos quarteis de Bombeiros nos Aeroportos de Cabo Verde, tem-se notado alguma carência.

Com a certificação dos Aeroportos na normativa ISO 9001 – Gestão da Qualidade, surgiram varias alertas no sentido de ter alguma aplicação para ajudar no tratamento de informações e mitigar possíveis perdas de dados que tem acontecido devido alta aglomeração de formulários em papel.

Contudo decidi avançar como TCC uma aplicação que fizesse a gestão da frota. No levantamento das necessidades para o início do desenvolvimento deparei com uma necessidade maior em relação aos extintores e cadastros. Com o andar do trabalho vi uma grande possibilidade na ampliação do aplicativo devido a potencialidade do Yii no desenvolvimento do mesmo.

Com isso o desafio e a motivação para um passo maior de desenvolver uma aplicação para tratamento geral de informações e não só a gestão das viaturas e manutenção.

1.5 Metodologia

Para desenvolver este projeto foi preciso várias pesquisas sobre aplicações web, engenharia de *softwares*, com o intuito de entender à especificação, desenvolvimento, manutenção e criação de sistemas de *software*, com aplicação de tecnologias e práticas de gestão de projetos e outras disciplinas, visando organização, produtividade e qualidade. Estas tecnologias e práticas englobam linguagens de programação, base de dados, ferramentas, plataformas, bibliotecas, padrões, processos e a questão da qualidade de *software*.

Os fundamentos científicos para a engenharia de *software* envolvem o uso de modelos abstratos e precisos que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de *software*, avaliando e garantindo suas qualidades. Além disso, a engenharia de *software* deve oferecer mecanismos para se planejar e gerenciar o processo de desenvolvimento de um sistema computacional.

Inicialmente foi feito uma avaliação do impacto de uma aplicação de gestão num quartel de Bombeiros, em que notou-se claramente que seria uma mais-valia e uma real necessidade atual.

Fez-se de seguida um segundo levantamento dos possíveis módulos a serem desenvolvidos para que realmente fosse abrangente e transversal a empresa.

Analisando o meio físico foram recolhidos dados dos equipamentos e viaturas para que os formulários fossem estandarizados e compatíveis.

O desenvolvimento deste sistema obrigou a aprofundar os conhecimentos na metodologia de desenvolvimento de aplicações web, permitindo a uma conclusão acima das minhas expetativas.

As dificuldades encontradas foram sendo ultrapassadas com uma boa orientação do orientador e muita persistência, focando no objetivo e nos prazos a cumprir.

1.6 Ferramentas utilizadas

Para desenvolvimento optei por usar o yii framework por ser uma das mais seguras na implementação de aplicações web. É também disponível gratuitamente tendo em conta que o custo das aplicações é muito levado em conta.

No Yii tudo é muito bem definido, as classes são de fácil extensão e como entende-las. As funcionalidades que não estão no Yii são disponíveis em extensões. Para mim a grande vantagem é o uso do PHP que é uma linguagem de fácil acesso mesmo para iniciantes em programação, é livre os custos de manutenção do PHP é muito reduzido bem como o custo de hospedagem.

O PHP integra na maioria das bases de dados da atualidade, é muito associado ao MYSQL mas pode rodar sem restrição no Oracle, MSSQL, IBM e outros com padrão OBDC.

O yii é adaptado par desenvolvimento em PHP que é uma linguagem de fácil acesso, que se pode encontrar no desenvolvimento da maioria das aplicações web.

Quanto a qualidade das aplicações web em PHP têm vindo a melhorar consideravelmente, principalmente com a entrada no mercado do yii, facilitando e reduzindo o tempo de elaboração de qualquer sistema.

1.7 Metodologia de pesquisa

Para que fosse desenvolvido o projeto foram delineadas certas medidas para alcançar os objetivos, com métodos claros. Como trabalho de campo, foram feitas entrevistas aos responsáveis do quarteis de Bombeiros dos Aeroportos a nível nacional, à alguns responsáveis de aeroportos e do gabinete de qualidade e colegas para recolha de dados, para que dessem algum *input* ao projeto.

NOME	CARGO	FUNÇÃO
José Lima Barber	Técnico Administrativo e Marketing	Diretor do Aeroporto Internacional Cesária Évora

Tiago Gonçalves	Técnico de Qualidade Higiene e Segurança no Trabalho	
Nelson Lima	Bombeiro	Chefe de Serviço de Bombeiros do Aeroporto Internacional Nelson Mandela
Amândio Almeida	Bombeiro	Chefe de Serviço de Bombeiros do Aeroporto Internacional Aristides Pereira
João do Rosário	Bombeiro	Supervisor de Turnos do Corpo de Bombeiros do Aeroporto Internacional Cesária Évora
Milton Lubrano	Bombeiro	Supervisor de Turnos do Corpo de Bombeiros do Aeroporto Internacional Cesária Évora
Anilton Andrade	Bombeiro	Presidente da Associação dos Bombeiros de São Vicente

Tabela 1 – Tabela de entrevistados

As entrevistas tiveram um papel preponderante no desenvolvimento do sistema, sendo que muitos das funcionalidades criadas no sistema forma ideias trabalhadas nas entrevistas.

Algumas propostas validas foram incluídas no projeto, também algumas que anteriormente estavam atribuídas no projeto como, tratamento e manutenção não eram necessárias tendo em conta haver uma aplicação transversal a empresa para tal fim.

Os entrevistados mostraram necessidade de ter um aplicativo que gere as informações dos quarteis bem como alguma logística, gestão de extintores, de forma organizada e concentrada num único canal.

Resultados esperados com o GestFrota

1.8 A nível tecnológico

Um dos objetivos estratégicos definidos pelo CdA da ASA no plano de atividade para 2016 foi a modernização dos serviços.

Com a necessidade de um aplicativo como o GestFrota, viu-se um meio de sincronizar o projeto TCC e a carência atual dos Aeroportos, para melhorar a comunicação e concentração das informações num único canal.

A organização dos extintores e atribuição de matrículas que serão facilmente monitorizadas pelo sistema.

Estandarizar a comunicação dentro do serviço, de modo que todos terão as mesmas informações.

Aumentar significativamente a rapidez no tratamento de informações.

Melhorar o cadastro dos equipamentos bem como a dos Bombeiros, facilitando no tratamento dos mesmo.

1.9 A nível Administrativo

A este nível o aplicativo terá um impacto maior, sendo que toda a gestão do quartel está dependente do gestor de serviço.

As informações serão transmitidas com maior rapidez e fluidez.

Toda a gestão feita pelo corpo diretivo do quartel será trabalhada logo que seja lançada no aplicativo.

A qualidade e eficiência do trabalho nos quarteis em todos os aeroportos terão um ganho considerável.

MÓDULO	FUNÇÃO
BOMBEIROS	O módulo Bombeiros tem como objetivo gerir toda a informação relativamente aos Bombeiros, desde o cadastro, a relatórios de turnos, os reportes. Com intuito claro de manter a base de dados atualizada em todas as atribuições com uma relação entre tabelas, facilitando uma leitura dinâmica de dados entre si.
VIATURAS	O módulo Viaturas tem como objetivo gerir toda a informação relativamente as Viaturas, desde o cadastro das viaturas, as avarias, a gestão do combustível. Com intuito claro de manter a base de dados atualizada em todas as atribuições, com uma relação entre tabelas, facilitando uma leitura dinâmica de dados entre si.
EMERGÊNCIA	A ser desenvolvido no futuro

Tabela 2 - Tabela dos Módulos

1.9.1 Módulo Bombeiros

Figura 2 - Fluxograma do Módulo Bombeiros

No Yii para que haja leitura dinâmica entre formulários, ou seja, formulário dinâmicos, cria-se os módulos e dentro dos módulos cria-se os modelos.

1.10 Descrição dos Modelos

Cadastro Bombeiro – Faz o cadastro do Bombeiro com toda a informação disponível que se achar necessário para a empresa;

Reporte – Faz o registo de todo o reporte que o colaborador se achar necessário. Também a gestão e listagem dos reportes na base de dados;

Bombeiros – Faz o registo dos Bombeiros que trabalham no Aeroporto para que as tabelas os formulários dinâmicos, de Check-List e Relatórios de Turnos, sejam chamados num único formulário através de um DropBox;

Relatório de Turno – Registo do trabalho efetuado em cada turno do dia, com ambiente dinâmico para chamar os Bombeiros que estejam registados no sistema.

1.10.1 Módulo Viaturas

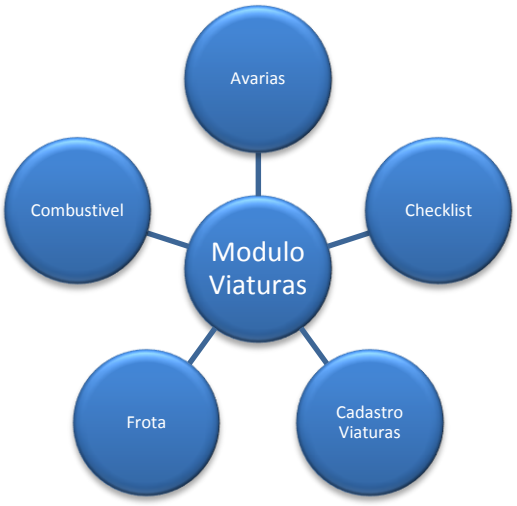


Figura 3 - Fluxograma do Módulo Viaturas

1.11 Descrição dos Modelos

Avarias – Faz o registo de todas as avarias nas viaturas, bem como a listagem. Uma avaria é considerada aberta até que seja reparada, sendo que no formulário é possível o registo de uma avaria sem que a data de reparação seja preenchida;

Check-List – Faz o registo do estado diário de todas as viaturas do quartel. É usado apenas um formulário que de forma dinâmica chama todas as viaturas inseridas no sistema através de um DropBox;

Combustível – Faz o registo do combustível recebido por cada viatura, em que podemos procurar os registos através do código de viaturas;

Cadastro Viaturas – Faz o cadastro de toda a informação pertinente das viaturas;

Frota – Faz o registo de todas as viaturas que estejam disponíveis no quartel.

1.11.1 Arquitetura do Sistema

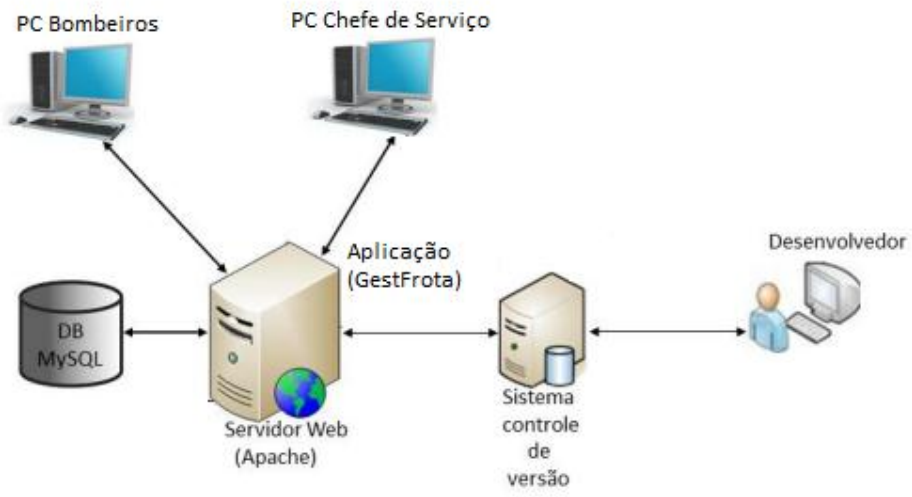


Figura 4 - Esquema da Arquitetura do Sistema

1.12 Trabalhos Relacionados

A necessidade de aplicações para a gestão de quarteis de Bombeiros tem sido cada vez mais, devido o grande fluxo de informação que é gerado nos serviços.

Nas situações de emergência dificilmente se consegue reter dados ou informações, sendo que o nível de *stress* esta muito alto, então a necessidade de aplicativos de registo e gestão de informações.

Dos mais usados é de referir os seguintes aplicativos:

- **IFFire;**(1)
- **SOS 193;**(2)
- **FireCast.**(3)

IFFire - uma aplicação desenvolvida exclusivamente para as Corporações de Bombeiros. É uma solução modular que cobre toda a área operacional e administrativa e que permite às Corporações de Bombeiros uma gestão mais moderna, mais organizada e mais eficiente.

Em termos comparativos, é de realçar alguns pontos em comum:

- Cadastro dos Bombeiros e Viaturas;

- Registo e tratamento de ocorrências do serviço;
- Tratamento e gestão dos reportes do serviço;

O aplicativo em termos de gestão é muito completo mas a GestFrota tem algumas funcionalidades que levam uma vantagem:

- Cadastro e gestão dos extintores;
- Registo e gestão das avarias;
- Apresentação dos dados do serviço em gráficos.

SOS 193 - o aplicativo SOS 193 pode ajudar a fazer os procedimentos iniciais de socorro até a chegada de equipe especializada. Tem como objetivo orientar a população em situações de emergência. O SOS 193 traz orientações sobre primeiros-socorros em emergências pré-hospitalares e afogamento.

Faz o registo das ocorrências em que os Bombeiros são chamados, e o cadastro dos artigos de socorro existentes na corporação.

FireCast - O FireCast é uma ferramenta para compartilhamento de informações sobre as solicitações de emergência de um Corpo de Bombeiros em atendimento. Disponível para *download* gratuito na loja virtual Google Play desde outubro de 2015, o aplicativo já foi instalado por mais de 1000 usuários. A versão 1.3.3 permite o compartilhamento das informações iniciais da ocorrência nas redes sociais de interesse. O usuário pode rapidamente compartilhar as informações por meio do *Whatsapp*, *Telegram*, correio eletrónico, *Hangouts*, *Twitter* entre outros.

Em relação a GestFrota pode ser identificado os seguintes pontos em comum:

- Tratamento de ocorrências e emergências;
- Tratamento de ocorrências do serviço.

O aplicativo GestFrota não tem as mesmas funcionalidades na gestão de ocorrências em emergências, mas sim em ocorrências do serviço, funcionalidades esta que pode ser desenvolvida num futuro próximo.

De todo há outras funcionalidades que a GestFrota esta um pouco mais a vontade no tratamento de dados tais como:

- Tratamento de avarias;
- Cadastro e gestão dos extintores;
- Cadastro dos Bombeiros e Viaturas;
- Exibição de dados de gestão do quartel.

2 CAPITULO II

2.1 Enquadramento do Projeto

As empresas e serviços com determinadas especificidades como à dos aeroportos de Cabo Verde precisam de aplicativos para a gestão das informações, devido o aumento do fluxo de dados.

A necessidade da organização, gestão dos dados do serviço dos Bombeiros do Aeroporto Internacional Cesária Évora fez com que aparecesse o aplicativo GestFrota, para melhorar e aumentar a rapidez no tratamento das informações.

Pelo mundo fora, viu-se a necessidade do desenvolvimento de aplicativos do tipo, sendo que ainda na maior parte dos quarteis pela Europa ainda não dispõem de tais ferramentas.

A cultura do desenvolvimento direcionado a gestão de logística de emergência, até mesma a gestão de crise começou nos Estados Unidos da América, devido a necessidade de resposta rápida, eficiente e estandardizada nas situações de emergência.

A implementação de este aplicativo nos aeroportos traz ganhos significativos, que podem ser medidos ou não, poupa nas tarefas e no tempo gasto, atendendo uma crescente necessidade de concentração de informação em um único aplicativo.

2.2 Impacto direto:

- Redução dos custos operacionais do serviço;
- Maior rapidez no tratamento dos Check-List:

- Aumento da produtividade dos colaboradores;
- Segurança no tratamento das informações;
- Armazenamento de dados de forma organizada.
- Imprimir maior qualidade no trabalho e de forma organizada;

Após a implementação do aplicativo, o tratamento das informações terá uma dinâmica muito maior da atual, favorecendo a gestão dos processos auditados, relacionados ao serviço de Bombeiros.

2.3 Framework Yii

Yii acrónimo para "Yes it is" é um *framework Open Source* de alta performance e escalabilidade em PHP, criado em 2008, que utiliza componentes para o desenvolvimento de grandes aplicações Web. Permite máxima reutilização de códigos na programação Web e pode acelerar significativamente o processo de desenvolvimento. O nome Yii (pronunciado i) representa as palavras fácil (*easy*), eficiente (*efficient*) e extensível (*extensible*).

Para executar uma aplicação Web que utilize o Yii, precisamos de um servidor Web com suporte a PHP 5.1.0 ou superior.

Para os desenvolvedores que desejam utilizar o Yii, é muito importante o conhecimento de programação orientada a objetos (POO), pois o Yii é um *framework* totalmente orientado a objetos.

O Yii é um *framework* de programação Web genérico que pode ser usado para desenvolver praticamente todos os tipos de aplicações Web. Por ser um *framework* leve equipado com sofisticadas soluções em *caching*, é especialmente adequado para o desenvolvimento de aplicações com alto tráfego de dados, tais como portais, fóruns, sistemas de gestão de conteúdo (CMS), sistemas de *e-Commerce*, etc.

Como a maioria dos *frameworks* PHP, O Yii é um *framework* MVC.

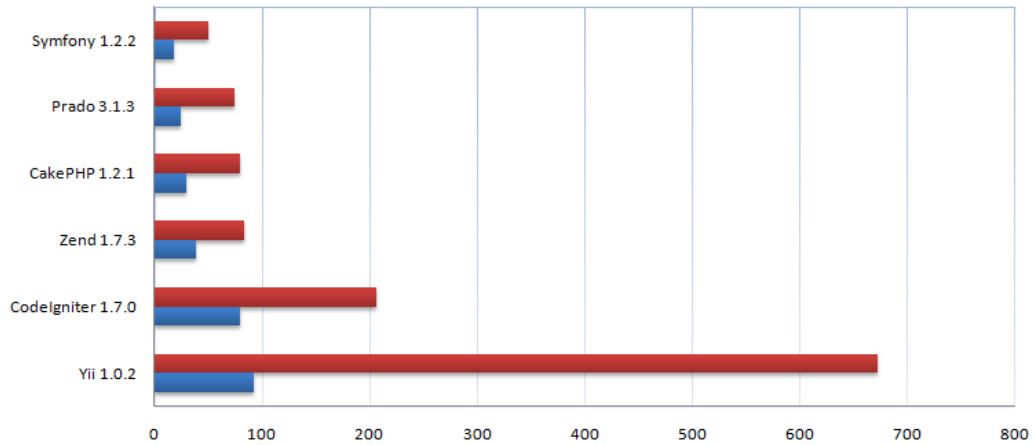
O Yii se sobressai dos outros *frameworks* PHP na medida em que é eficiente, rico em recursos e bem documentado. O Yii é cuidadosamente projetado para ajustar a sérias aplicações Web desde seu início. [(4)]

2.4 Vantagens do Yii a outros framework

É um excelente suporte a MVC, DAO / *Active Record*, I18N (*Internationalization*) / L10N (*Localization*), Cache, Autenticação e Controle de acesso (RBAC), Testes unitários, Gerador de códigos automáticos, *Skinning and Theming*, *Jquery* / solicitações Ajax integrado aos *widgets* do Yii, *Web Services* (WSDL) e muito mais.

A arquitetura permite que o mesmo carregue somente o necessário para aplicação no presente momento e, em conjunto com o suporte a cache (APC), consegue um RPS (Requisição por Segundo).

2.4.1 Comparação de Desempenho dos Framework de PHP



	Yii 1.0.2	CodeIgniter 1.7.0	Zend 1.7.3	CakePHP 1.2.1	Prado 3.1.3	Symfony 1.2.2
RPS with APC	673	206	83	79	75	50
RPS without APC	93	79	38	29	24	18

Figura 5 - Gráfico de Comparação do Desempenho dos Framework [5]

2.5 Segurança

Por padrão faz o tratamento/validações de entrada e saída de dados, oferece suporte a ataques SQL *Injection*, Cross-site *scripting* (XSS), CSRF (*Cross-Site Request Forgery*), *Cookie Attack* e entre outros.

Aceita muito bem integrações com códigos de terceiros, como por exemplo, *PEAR*, *Codeigniter*, *Zend Framework.*

Yii segue o padrão MVC em sua estrutura, garantindo a separação entre camadas lógicas e camadas de apresentação do projeto a ser desenvolvido. Auxilia o desenvolvedor a ter um código mais limpo e reutilizável (DRY – *Don't repeat yourself*) sem grandes esforços, supre a elaboração de sites simples bem como aplicações extremamente complexas, sendo necessário preocupar apenas com tarefas específicas do projeto. (6)

2.6 Características do Yii

OOP, MVC, DRY, DAO/QueryBuilder/AR Form Input Validation, AJAX Widgets, RBAC, Skin and Theme, WebService, I18N, L10N, Layered Cache, Error handling and log, segurança (CSRF, XSS, SQL Injection, HTML Purify and Cookie), teste unitario e functional, XHTML, extensões, modularidade, Drive Event, Scaffolding. (7)

2.7 Versões do Yii (8)

Do início do projeto Yii, em Janeiro de 2008 por Quiang Xue, as novidades no aplicativo não param de crescer, impondo um ritmo de crescimento acelerado para acompanhar a necessidade dos desenvolvedores.

Desde o início até dias de hoje o Yii já teve 11 versões:

- Versão 1.1.0
- Versão 1.1.1
- Versão 1.1.2
- Versão 1.1.3
- Versão 1.1.4
- Versão 1.1.5
- Versão 1.1.6
- Versão 1.1.7
- Versão 1.1.8
- Versão 1.1.9
- Versão 2.0

2.7.1 Versão 1.1.0

- Adicionado suporte para escrever testes unitários.
- Adicionado suporte para usar *skins* em *widget*.
- Adicionado um construtor de formulários extensível.
- Melhorado a forma de declarar atributos seguros no modelo. *Securing Attribute Assignments.*
- Modificado o padrão do algoritmo de carregamento relacional em consultas *Active Record*, de modo que todas as tabelas são unidas em uma única instrução SQL.
- Modificado o alias padrão de tabelas para ser o nome do relacionamento *active record.*
- Adicionado suporte para uso de prefix em tabelas.
- Adicionado um conjunto de novas extensões conhecido como o *Zii library*.
- O nome do alias para a tabela principal em uma consulta AR é fixado para ser 't'.

2.7.2 Versão 1.1.1

- Adicionado *CActiveForm* que simplifica a escrita de códigos de formulários com suporte a validação de dados em ambos aos lados, cliente e servidor;
- Refeito o código gerado pela ferramenta yiic. Em particular, o esqueleto da aplicação é agora gerado com vários *layouts*, o menu de operação foi reorganizado para páginas CRUD; adicionado recurso de pesquisa e filtragem para a página admin gerada pelo comando CRUD; usado *CActiveForm* para renderizar o formulário;
- Adicionado suporte para permitir definição global de comandos yiic.

2.7.3 Versão 1.1.2

- Adicionada ferramenta Web para geração de código chamada Gii.

2.7.4 Versão 1.1.3

- Adicionado suporte para configurar valores padrões em *widgets* na configuração da aplicação.

2.7.5 Versão 1.1.4

- Adicionado suporte para automática ligação de parâmetros de ações.

2.7.6 Versão 1.1.5

- Adicionado suporte para as ações e parâmetros de ação nos comandos do console;
- Adicionado suporte para *autoloading* de classes com *namespace;*
- Adicionado suporte para temas em widget com *views.*

2.7.7 Versão 1.1.6

- Adicionado *query builder;*
- Adicionado *database migration;*
- Melhores práticas MVC;
- Adicionado suporte para o uso de parâmetros anônimos e opções globais em comandos do console.

2.7.8 Versão 1.1.7

- Adicionado suporte para URL RESTful;
- Adicionado suporte de cache para consultas;
- Agora é possível passar parâmetros para uma requisição relacional;
- Adicionado a capacidade para realizar consultas relacionais sem ter modelos relacionados;
- Adicionado suporte para HAS_MANY através de HAS_ONE com relacionamento AR;
- Adicionado suporte para transações no recurso de migração de banco de dados;

- Adicionado suporte para a utilização de parâmetros de ligação com *class-based actions*;
- Adicionado suporte para validação de dados no cliente com CActiveForm.

2.7.9 Versão 1.1.8

Adicionando suporte para o uso customizado de regras URLs em classes.

2.7.10 Versão 1.1.9

- Adicionado cerca de 60 melhoras de correção de Bugs.

2.7.11 Versão 2.0

- Esta versão inclui centenas de novas funcionalidades, alterações e correções de *bugs* que serão feitas ao longo do lançamento das próximas versões 2.XX.

2.8 Arquitetura do Yii

Para facilitar os desenvolvedores no tratamento de formulários, uma das principais tarefas no desenvolvimento, os criadores do Yii integraram a arquitetura MVC (*Model View Controller*).

A arquitetura MVC consiste na separação lógica das principais camadas da aplicação, desde a camada mais abstrata até a mais concreta. A proposta de separar as camadas parece muito fácil em sua teoria, entretanto, colocá-la em prática não é tão simples.

Sendo assim, os criadores do MVC padronizaram sua divisão basicamente em três camadas: modelo (*model*), visualização (*view*) e controlador (*controller*). [(9)]

2.8.1 Camada de Modelo (Model):

- Este é o local onde ficam todas as regras de negócio, ou seja, onde é feita a abstração dos dados. Aqui é definido **a forma e as restrições dos dados** os quais estarão disponíveis na aplicação. O modelo interage diretamente com as *views* e os controladores (camadas de visualização e controle, respetivamente). O modelo comunica a estes outros dois componentes quaisquer alterações feitas, com isso os controladores manipulam os dados e as visualizações renderizam. [(10)]

2.8.2 Camada de Controlo (Controller):

- A camada de controlo é responsável por manipular os dados vindos do modelo (*model*), além de enviá-los, possuindo alterações ou não de volta à camada de visualizações (*view*).
  Uma vez recebido o dado a ser manipulado, o controlador executa ações (*actions*) com estes dados. Na prática estas ações (*actions*) são métodos que, depois de executada sua função, irão chamar uma *view* passando os dados manipulados pelo controlador. (11)

2.8.3 Camada de Visualização (View):

- Esta é a camada responsável por exibir os dados para o usuário final, ela possui comunicação direta com o controlador. **As visualizações não possuem nenhuma ligação direta com a camada de Modelos.** (12)

2.9 MVC no PHP (13)

Diferentemente de outras linguagens, como o C# através da plataforma .NET, o PHP possui algumas "dificuldades" para se adaptar ao MVC. "Dificuldades" pois, comparado às aplicações em C# que só rodam utilizando o framework .NET, o PHP não possui nenhum framework base para rodar suas aplicações. Além disso, há o fato de o PHP ser interpretado, e não compilado, como nas linguagens que utilizam o framework .NET.

Por isso, para se adequar ao **MVC** no **PHP**, é necessário criar suas estruturas de diretórios manualmente, além de criar adaptações para haver a interação entre as três principais camadas do MVC.

O Yii implementa o padrão de desenvolvimento modelo-visão-controle (MVC) que é amplamente adotado na programação Web. O MVC visa separar a lógica de negócio da interface com o usuário, assim os programadores podem mudar facilmente cada parte, sem afetar as outras. No padrão MVC, o modelo representa as informações (os dados) e as regras de negócio, a visão contém elemento de interface com o usuário, como textos, formulários, e o controlo gere a comunicação entre o modelo e a visão.

Além MVC, o Yii também introduz um controlo de frente, chamada aplicação (*application*), que representa o contexto de execução dos processos requisitados. A

aplicação recebe a solicitação do usuário e a envia para um controlador adequado para ser processada.

O diagrama seguinte mostra a estrutura estática de uma aplicação Yii:

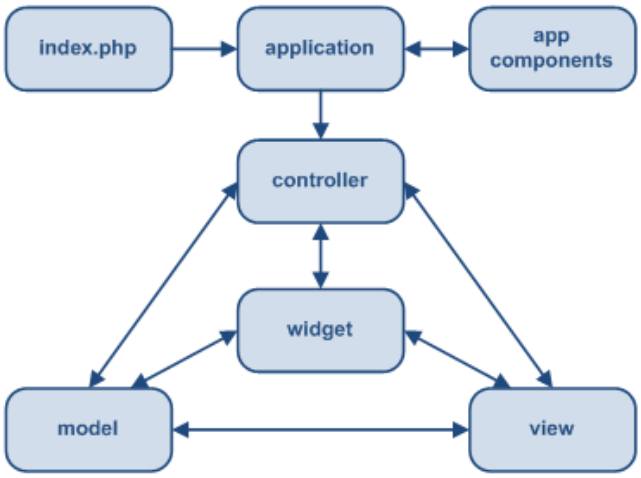


Figura 6 - Diagrama da Estrutura Estática de App no Yii [14]

2.10 Estrutura de diretórios

Confira na figura 7 a ilustração com a estrutura em árvore dos diretórios da aplicação "TCC (GestFrota) " e a seguir uma breve explicação sobre ela.

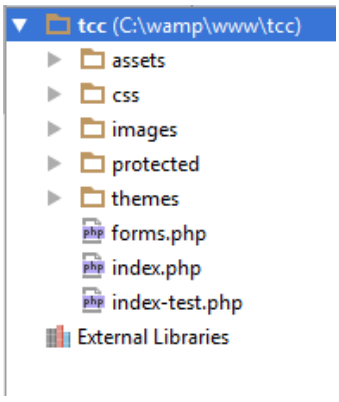


Figura 7 - Esquema da Estrutura de Diretórios

2.10.1 Pasta Assets:

- É a pasta responsável por armazenar todos os recursos adicionais que a aplicação necessita. São guardados arquivos *JavaScript* (como *jQuery*), arquivos de folhas

de estilo CSS para a estilização de *Widgets* do *framework*, além de algumas imagens que foram utilizadas no aplicativo.

2.10.2 Pasta CSS:

- Aqui ficarão guardadas todos os arquivos folhas de estilo do aplicativo.

Pasta Images: Diretório onde se localizam todas as imagens utilizadas na aplicação.

2.10.3 Pasta Protected:

- Pasta responsável por configurar todo o lado servidor da aplicação.

2.10.4 Pasta Themes:

- Arquivos que posteriormente serão utilizados como *template* da aplicação.

2.11 Componentes do Yii

As aplicações feitas com o Yii são construídas por componentes, que são objetos desenvolvidos para um fim específico.

A funcionalidade do objeto da aplicação pode ser facilmente customizada e enriquecida usando a arquitetura flexível de componentes. O objeto gere um conjunto de componentes, que implementam recursos específicos. Por exemplo, realiza algum processamento inicial da solicitação do usuário com a ajuda dos componentes CUrlManager e CHttpRequest.

Um componente é uma instância da classe *CComponent*, ou uma de suas classes derivadas. A utilização de um componente basicamente envolve o acesso a suas propriedades e a execução/manipulação de seus eventos. A classe base *CComponent* especifica como definir propriedades e eventos.

Ao configurar as propriedades dos componentes da instância da aplicação, podemos personalizar a classe e os valores das propriedades de qualquer componente usado na aplicação. Por exemplo, podemos configurar o componente CMemCache para que ele possa utilizar múltiplos servidores de *memchache* para fazer o *caching*. [(15)]

2.12 Propriedade de um Componente

Uma propriedade de um componente é como uma variável membro pública de um objeto. Nós podemos ler seu conteúdo ou lhe atribuir novos valores.

```
$width=$component->textWidth;      // acessa a propriedade textWidth
$component->enableCaching=true;    // altera a propriedade enableCaching
```

Para definir uma propriedade em um componente, podemos simplesmente declarar uma variável membro pública na classe do componente. No entanto, existe uma maneira mais flexível, que consiste em definir métodos assessores (getters e setters), como no exemplo a seguir: (15)

```
public function getTextWidth()
{
    return $this->_textWidth;
}

public function setTextWidth($value)
{
    $this->_textWidth=$value;
}
```

2.13 Principais Componentes da Aplicação (16)

O Yii predefine um conjunto de componentes principais da aplicação para fornecer funcionalidades comuns em aplicações Web. Por exemplo, o componente *request* é usado para coletar informações sobre uma solicitação do usuário e prover informações como a URL solicitada e *cookies*. Ao configurar as propriedades desses componentes principais, podemos mudar o padrão de comportamento de praticamente todos os aspetos do Yii.

Abaixo, a lista dos principais componentes que são pré-declarados pelo CWebApplication.

- **assetManager: CAssetManager** - gere a criação dos ativos privados (*assets*).
- **authManager: CAuthManager** - gere o controlo de acesso baseado em regras (RBAC).

- **cache: CCache** - fornece as funcionalidades do *caching* de dados. Deve-se especificar a classe atual (ex.: CMemCache, CDbCache). Caso contrário, será retornado *Null* ao ter acesso ao componente.
- **clientScript: CClientScript** - gere os *scripts* (javascript e CSS) do cliente.
- **coreMessages: CPhpMessageSource** - fornece as principais mensagens traduzidas usadas pelo framework Yii.
- **db: CDbConnection** - fornece uma conexão ao banco de dados. Deverá ser configurado a propriedade connectionString corretamente para que seja utilizado.
- **errorHandler: CErrorHandler** - processa erros e exceções do PHP.
- **format: CFormatter** - formata valores de dados com o propósito de exibi-los.
- **messages: CPhpMessageSource** - fornece mensagens traduzidas utilizadas pela aplicação Yii.
- **request: CHttpRequest** - fornece informações relacionadas à solicitação do usuário.
- **securityManager: CSecurityManager** - fornece serviços relacionados à segurança, como *hashing* e encriptação.
- **session: CHttpSession** - fornece funcionalidades relacionadas à sessão.
- **statePersister: CStatePersister** - fornece o mecanismo para a persistência do estado global.
- **urlManager: CUrlManager** - fornece funcionalidades de análise e criação de URLs.
- **user: CWebUser** - guarda informações relacionadas à identidade sobre o usuário atual.
- **themeManager: CThemeManager** - gere temas.

2.14 Workflow – Fluxo de Trabalho [17]

Os fluxogramas demostrados na figura 8 mostram o fluxo de trabalho do desenvolvimento da aplicação web utilizando o Yii. Este fluxo assume que já foi realizado a análise de requisitos, bem como a análise de *design* para a aplicação.

O diagrama a seguir mostra um típico fluxo de execução de uma aplicação Yii quando esta está recebendo uma solicitação de um usuário.

Fluxo de execução da aplicação Yii

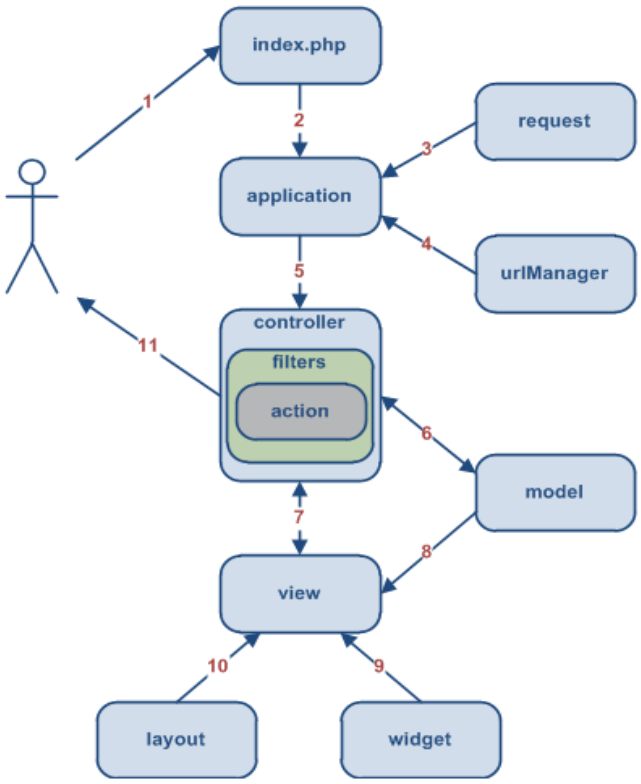


Figura 8 – Fluxograma de Execução da Aplicação [17]

1. O usuário faz o pedido de uma solicitação e o servidor Web processa executando o *script bootstrap* index.php através do URL:

   http://www.exemplo.com/index.php?r=post/show&id=1 .

2. O *script* de *bootstrap* cria uma instancia de aplicação (*application*) e a executa.

3. A aplicação obtém as informações detalhadas da solicitação de um componente da aplicação chamado *request*.
4. A aplicação determina o controle e a ação requerida com a ajuda do componente chamado *urlManager*. Para este exemplo, o controle é *post* que se refere à classe PostController e a ação é show cujo significado real é determinado no controlo.
5. A aplicação cria uma instância do controle solicitado para poder lidar com a solicitação do usuário. O controlo determina que a ação show refere-se a um método chamado actionShow no controle da classe. Em seguida, cria e executa filtros (por exemplo, o controle de acesso, *benchmarking*) associados a esta ação. A ação só é executada se permitida pelos filtros.
6. A ação lê um modelo Post cujo ID é `1` no Banco de Dados.
7. A ação processa a visão chamada show, com o Post.
8. A visão apresenta os atributos do modelo Post.
9. A visão executa alguns *Widgets*.
10. O resultado do processamento da visão é embutido em um *layout*.
11. A ação conclui o processamento da visão e exibe o resultado ao usuário.

2.15 Ciclo de Vida de uma Aplicação [18]

Quando processa uma solicitação de um usuário, a aplicação segue o ciclo de vida descrito a seguir:

1. Pré-inicia a aplicação com o método CApplication::preinit();
2. Configura o Auto carregamento de classes (*autoloader*) e o tratamento de erros;
3. Regista os principais componentes da aplicação;
4. Carrega as configurações da aplicação;
5. Inicia a aplicação com CApplication::init():
6. Regista os comportamentos (*behaviors*) da aplicação;
7. Carrega os componentes estáticos da aplicação;

8. Dispara um evento onBeginRequest (no início da requisição);
9. Processa a solicitação do usuário:
10. Coleta informações sobre a solicitação do usuário;
11. Cria um controlo;
12. Executa o controlo;
13. Dispara um evento onEndRequest (ao fim da requisição).

3 CAPITULO III

3.1 Engenharia de Software

Ao passar do tempo, o *software* esta a consolidar como um elemento muito importante para o mundo e aumentando a capacidade de manipular a informação. Como muitos elementos computacionais tiveram mudanças até hoje e continuam tendo. Com este crescimento computacional, levam a criação de sistemas perfeitos e problemas para quem desenvolvem *softwares* complexos. As preocupações dos engenheiros de *software* para desenvolverem os *softwares* sem defeitos e entregarem estes produtos no tempo marcado, assim leva a aplicação da disciplina de engenharia de *software*. Com o crescimento desse segmento muitas empresas possuem mais especialistas em TI em que cada um tem sua responsabilidade no desenvolvimento de *software* e é diferente de antigamente que era um único profissional de *software* que trabalhava sozinho numa sala [PRESSMAN, 2006].

No ponto a frente será demostrado o modelo escolhido para o desenvolvimento do sistema.

3.2 Sistema de Informação

Um sistema de informação é um conjunto organizado de elementos, podendo ser pessoas, dados, atividades ou recursos materiais em geral. Estes elementos interagem entre si para processar informação e divulga-la de forma adequada em função dos objetivos de uma organização.

O estudo dos sistemas de informação surgiu como uma subdisciplina das ciências da computação, com o objetivo de racionalizar a administração da tecnologia no seio das organizações. O campo de estudo foi-se desenvolvendo até vir mesmo a fazer parte dos estudos superiores dentro da administração.

No prisma empresarial, os sistemas de informação podem classificar-se de diversas formas. Existem, sistemas de informação operacional ou de processamento de transações (que gerem a informação referente às transações numa empresa), sistemas de informação de gestão (para solucionar problemas empresariais em geral), sistemas de informação estratégicos ou de apoio à decisão (analisam as distintas variáveis de negócio para a tomada de decisões), sistemas de informação executiva (para diretores, gerentes e administradores), sistemas de automatização de escritórios (aplicações que ajudam no trabalho administrativo) e sistemas especializados (que emulam o comportamento de um especialista numa área concreta). (19)

Componentes de um sistema de informação

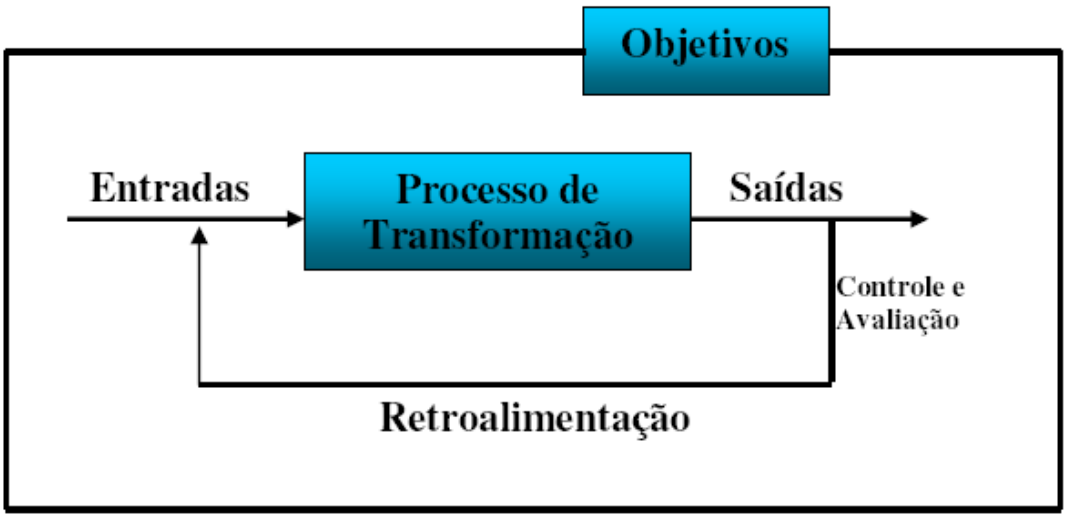


Figura 9 - Diagrama dos Componentes do Sistema

3.3 Enquadramento do Aplicativo no Sistema de Informação

Os sistemas de informação podem ser classificados de acordo a informação que será processada. Desta forma, a classificação dos sistemas de informação geralmente é feita

de acordo com a pirâmide empresarial, composta pelos níveis estratégicos das organizações.

Dos sistemas de informação referidos, o melhor que se enquadra no aplicativo desenvolvido é o Sistema de Informação Operacional – SIO.

Formado por operações rotineiras; normalmente trabalha com um grande volume de operações de entrada e saída.

Exemplos: formulários de cadastros, relatórios de turno, listagens, consultas e modificações de dados entre outros.

3.4 Fluxo de tomada de decisão

Durante o desenvolvimento do aplicativo, com base nos métodos do sistema de informação operacional, foi seguido os requisitos conforme fluxo abaixo.

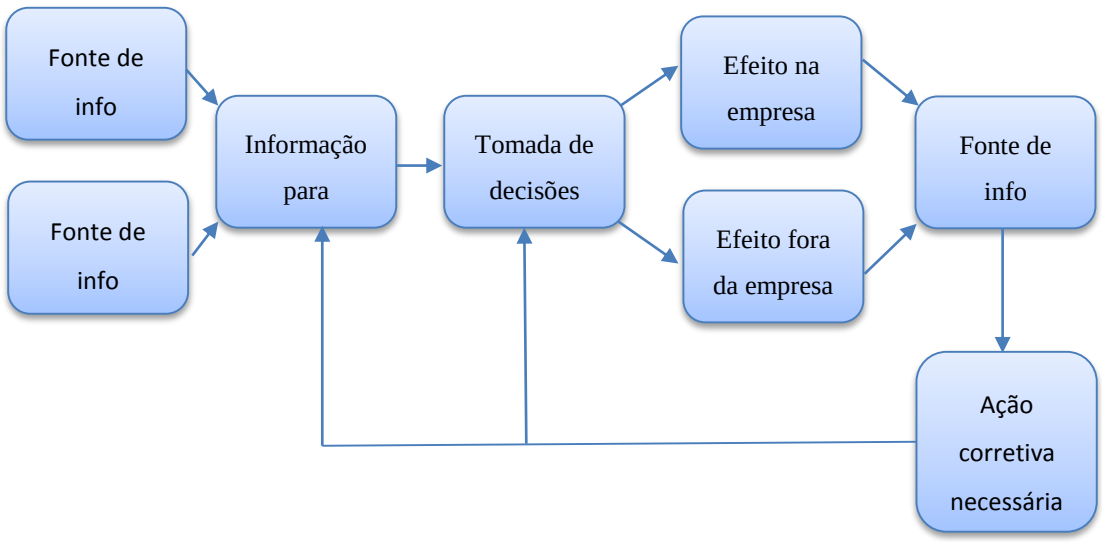


Figura 10 – Fluxo da tomada de decisões

3.5 Modelo Cascata

O modelo em cascata é um modelo de desenvolvimento de software sequencial no qual o desenvolvimento é visto como um fluir constante para frente (como uma cascata) através das fases de análise de requisitos, projeto, implementação, testes (validação), integração, e manutenção de *software*. A origem do termo cascata é frequentemente citado como

sendo um artigo publicado em 1970 por W. W. Royce; ironicamente, Royce defendia um abordagem iterativa para o desenvolvimento de *software* e nem mesmo usou o termo cascata. Royce originalmente descreve o que é hoje conhecido como o modelo em cascata como um exemplo de um método que argumentava ser um risco e um convite para falhas.

Para seguir um modelo em cascata, o progresso de uma fase para a próxima se dá de uma forma puramente sequencial. Inicialmente completa-se a especificação de requisitos, elaborando um conjunto rígido de requisitos do *software*, embora as especificações dos requisitos reais sejam mais detalhados, em um procedimento para projeto. Quando e somente o projeto está terminado, uma implementação para este projeto é feita pelos codificadores. Encaminhando-se para o próximo estágio da fase de implementação, inicia-se a integração dos componentes de *software*. (20)

Modelo Cascata

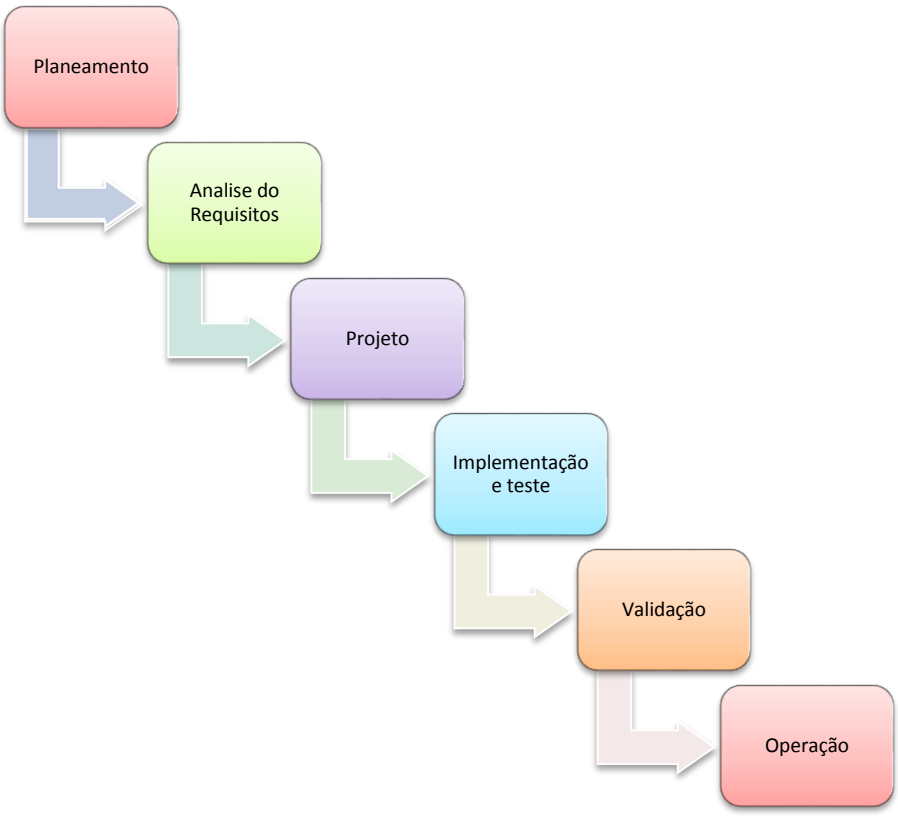


Figura 11 - Fluxo do Modelo Cascata

3.6 Analise dos requisitos do modelo

Nesta etapa, estabelecem-se os requisitos do *software* que se deseja desenvolver, o que consiste usualmente nos serviços que se devem fornecer, limitações e objetivos do *software*. Sendo isso estabelecido, os requisitos devem ser definidos de uma maneira apropriada para que sejam úteis na etapa seguinte. Esta etapa inclui também a documentação e o estudo da facilidade e da viabilidade do projeto com o fim de determinar o processo de início de desenvolvimento do projeto do sistema; pode ser vista como uma conceção de um produto de *software* e também como o início do seu ciclo de vida.

3.6.1 Planeamento

No planeamento do desenvolvimento do aplicativo, para atingir o objetivo na construção do sistema foi preciso elaborar alguns pontos para organizar:

Criar as tabelas no banco de dados;

Criar as classes do modelo para permitir que o aplicativo interaja com as tabelas do banco de dados;

Criar as classes controladoras, onde colocaremos as funcionalidades;

Criar apresentação.

3.6.2 Analise dos requisitos

Trata-se de uma fase fundamentalmente de engenharia e modelação do sistema de informação. O *software* é sempre o subconjunto de um conjunto (sistema) maior; por isso, o trabalho inicial consiste no estabelecimento das necessidades e requisitos globais de informação da organização e na subsequente atribuição de um subconjunto ao *software*. O processo de determinação dos requisitos é, de seguida, intensificado, concentrando-se especificamente no *software* a construir. É necessário que os engenheiros de sistemas (analistas) compreendam bem o domínio aplicacional nas suas diversas vertentes – informação, funções, comportamento, desempenho e interfaces.

3.6.3 Projeto

Denominada igualmente de desenho lógico, visa modelar uma solução de um modo independente da tecnologia, isto é, baseada nas funções do sistema a informatizar e não na tecnologia em que o sistema será implementado. O desenho dos programas traduz os requisitos num modelo de *software* cuja qualidade pode ser avaliada antes de se proceder à codificação. A transformação de um projeto para um código deve ser a parte mais evidente do trabalho da engenharia de *software*. O desenho físico tem de ser traduzido para uma linguagem própria da máquina.

3.6.4 Implementação e teste

É a fase do processo de teste de software e de hardware em que o sistema já completamente integrado é verificado quanto a seus requisitos num ambiente de produção. Está no escopo da técnica de teste de caixa-preta, e dessa forma não requer conhecimento da estrutura (lógica) interna do sistema. É um teste mais limitado em relação aos testes de unidade e de integração, fases anteriores do processo de teste, pois se preocupa somente com aspetos gerais do sistema.

O teste de sistema não se limita a testar somente requisitos funcionais, mas também requisitos não funcionais como a expectativa do cliente, e por isso inclui também técnicas não funcionais de teste.

3.6.5 Validação

O objetivo da Validação e da Verificação é assegurar que o aplicativo seja adequado e se atende às necessidades, ou seja, a confirmação de que este cumpra suas especificações.

A Verificação é uma atividade, a qual envolve a análise de um sistema para certificar se este atende aos requisitos funcionais e não funcionais. Já a Validação, é a certificação de que o sistema atende as necessidades e expectativas do cliente. O processo de Validação e Verificação, não são processos separados e independentes.

Durante este processo, podem ser utilizadas outras técnicas em conjunto, como a Inspeção de aplicativo. A Inspeção de um aplicativo é o ato de analisar as representações do sistema, analisar os documentos de requisitos, diagramas e código-fonte. É recomendável a utilização da Inspeção em todas as etapas do processo. Análise automatizada é uma

técnica estática, a qual não se torna necessária à execução do aplicativo. Testes de aplicativo executam o sistema utilizando uma massa de dados. Os dados são inseridos no sistema e sua saída é analisada. Esses testes servem para verificar se o desempenho do sistema está correto. O Processo de V&V liga todo o ciclo de vida do sistema. A Técnica V&V deve estabelecer a confiança de que o Sistema é adequado ao seu propósito. Deve atender as necessidades dos seus usuários e que seja útil para a tarefa pretendida.

3.6.6 Operação

Uma importante tarefa antes da operação é a documentação do projeto interno do aplicativo para propósitos de futuras manutenções e aprimoramentos. As documentações mais importantes são das interfaces externas.

Uma grande percentagem dos projetos de aplicativo falham pelo fato de o desenvolvedor não perceber que não importa quanto tempo a equipa de planeamento e desenvolvimento irá trabalhar na criação do aplicativo se ninguém da organização irá usá-lo. As pessoas ocasionalmente resistem à mudança e evitam aventurar-se em áreas pouco familiares. Como parte da fase de desenvolvimento, é muito importante o treinamento para os usuários de aplicativos mais entusiasmados, alternando o treinamento entre usuários neutros e usuários favoráveis ao aplicativo.

A manutenção e a melhoria do aplicativo fazem parte da necessidade e da descoberta de novos itens, necessidades ou erros no aplicativo.

3.7 Ferramentas utilizadas

3.7.1 PHP

É uma linguagem interpretada livre, usada originalmente apenas para o desenvolvimento de aplicações presentes e atuantes no lado do servidor, capazes de gerar conteúdo dinâmico na *World Wide Web*. Figura entre as primeiras linguagens passíveis de inserção em documentos HTML, dispensando em muitos casos o uso de arquivos externos para eventuais processamentos de dados. O código é interpretado no lado do servidor pelo módulo PHP, que também gera a página web a ser visualizada no lado do cliente. A linguagem evoluiu, passou a oferecer funcionalidades em linha de comando, e além disso,

ganhou características adicionais, que possibilitaram usos adicionais do PHP, não relacionados a *web sites*. [21]

3.7.2 HTML

É a sigla de **HyperText Markup Language**, expressão inglesa que significa "Linguagem de Marcação de Hipertexto". Consiste em uma linguagem de marcação utilizada para produção de páginas na web, que permite a criação de documentos que podem ser lidos em praticamente qualquer tipo de computador e transmitidos pela internet.

Para escrever documentos HTML não é necessário mais do que um editor de texto simples e conhecimento dos códigos que compõem a linguagem. Os códigos (conhecidos como tags) servem para indicar a função de cada elemento da página Web. Os tags funcionam como comandos de formatação de textos, formulários, links (ligações), imagens, tabelas, entre outros.

Os navegadores (browsers) identificam as *tags* e apresentam a página conforme está especificada. Um documento em HTML é um texto simples, que pode ser editado no Bloco de Notas (Windows) ou Editor de Texto (Mac) e transformado em hipertexto.

A linguagem HTML foi criada por Tim Barners Lee na década de 1990. As especificações da linguagem são controladas pela W3C (World Wide Web Consortium). [22]

3.7.3 JQuery

JQuery é uma biblioteca de código aberto e possui licença dual, fazendo uso da Licença MIT ou da GNU General Public License versão 2. A sintaxe do jQuery foi desenvolvida para tornar mais simples a navegação do documento HTML, a seleção de elementos DOM, criar animações, manipular eventos e desenvolver aplicações AJAX. A biblioteca também oferece a possibilidade de criação de plugins sobre ela. Fazendo uso de tais facilidades, os desenvolvedores podem criar camadas de abstração para interações de mais baixo nível, simplificando o desenvolvimento de aplicações web dinâmicas de grande complexidade. [23]

3.7.4 CSS

Cascading Style Sheets (CSS) é uma linguagem de folhas de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML. O seu principal benefício é a separação entre o formato e o conteúdo de um documento.

Em vez de colocar a formatação dentro do documento, o desenvolvedor cria uma ligação (Link) para uma página que contém os estilos, procedendo de forma idêntica para todas as páginas de um portal. Quando quiser alterar a aparência do portal basta modificar apenas um arquivo.

Com a variação de atualizações dos navegadores (browsers) como Internet Explorer que ficou sem nova versão de 2001 a 2006, o suporte ao CSS pode variar. O Internet Explorer 6, por exemplo, tem suporte total a CSS1 e praticamente nulo a CSS2. Navegadores mais modernos como Google Chrome e Mozilla Firefox tem suporte maior, inclusive até a CSS3, ainda em desenvolvimento. (24)

3.8 Base de Dados

3.8.1 MySQL

O MySQL é um sistema de gestão de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo. Entre os usuários do banco de dados MySQL estão: NASA, Friendster, Banco Bradesco, Dataprev, HP, Nokia, Sony, Lufthansa, U.S Army, US. Federal Reserve Bank, Associated Press, Alcatel, Slashdot, Cisco Systems, Google CanaVialis S.A. e outros.

Criado na Suécia por dois suecos e um finlandês: David Axmark, Allan Larsson e Michael "Monty" Widenius, que têm trabalhado juntos desde a década de 1980. Hoje seu desenvolvimento e manutenção empregam aproximadamente 400 profissionais no mundo inteiro, e mais de mil contribuem testando o *software*, integrando-o a outros produtos.

O sucesso do MySQL deve-se em grande medida à fácil integração com o PHP incluído, quase que obrigatoriamente, nos pacotes de hospedagem de *sites* da Internet oferecidos atualmente. (25)

3.8.2 SQL

SQL é sigla inglesa de "**Structured Query Language**" que significa, em Português, Linguagem de Consulta Estruturada, uma linguagem padrão de gerenciamento de dados que interage com os principais bancos de dados baseados no modelo relacional.

Alguns dos principais sistemas que utilizam SQL são: MySQL, Oracle, Firebird, Microsoft Access, PostgreSQL (código aberto), HSQLDB (código aberto e escrito em Java).

A linguagem SQL surgiu em 1974 e foi desenvolvida nos laboratórios da IBM como interface para o Sistema Gerenciador de Banco de Dados Relacional (SGBDR) denominado SYSTEM R. Esse sistema foi criado com base em um artigo de 1970 escrito por Edgar F. Codd.

Outras linguagens do gênero surgiram, mas a SQL tornou-se a mais utilizada. A criação de um padrão para a SQL foi realizada em 1986 pelo **American National Standard Institute (ANSI)** e em 1987 pela **International Organization for Standards (ISO)**.

SQL é uma linguagem essencialmente declarativa. Isso significa que o programador necessita apenas indicar qual o objetivo pretendido para que seja executado pelo SGBDR.

Alguns dos principais comandos SQL para manipulação de dados são: INSERT (inserção), SELECT (consulta), UPDATE (atualização), DELETE (exclusão). SQL possibilita ainda a criação de relações entre tabelas e o controle do acesso aos dados. [26]

3.8.3 Apache

O **servidor Apache** (ou Servidor HTTP Apache, em inglês: *Apache HTTP Server*, ou simplesmente: **Apache**) é o mais bem-sucedido servidor web livre. Foi criado em 1995 por Rob McCool, então funcionário do NCSA (National Center for Supercomputing Applications). Em uma pesquisa realizada em dezembro de 2007, foi constatado que a utilização do Apache representa cerca de 47.20% dos servidores ativos no mundo. Em maio de 2010, o Apache serviu aproximadamente 54,68% de todos os *sites* e mais de 66% dos milhões de sites mais movimentados. É a principal tecnologia da Apache *Software Foundation*, responsável por mais de uma dezena de projetos envolvendo tecnologias de transmissão via web, processamento de dados e execução de aplicativos distribuídos.

O servidor é compatível com o protocolo HTTP versão 1.1. Suas funcionalidades são mantidas através de uma estrutura de módulos, permitindo inclusive que o usuário escreva seus próprios módulos — utilizando a API do *software*. (27)

3.8.4 Características do servidor Apache (28)

O Apache *Server* é um software livre, o que significa que qualquer um pode estudar ou alterar seu código-fonte, além de poder utilizá-lo gratuitamente. É graças a essa característica que o *software* foi (e continua sendo) melhorado ao passar dos anos. Graças ao trabalho muitas vezes voluntário de vários desenvolvedores, o Apache continua sendo o servidor Web mais usado no mundo.

Além de estar disponível para o Linux (e para outros sistemas operacionais baseados no Unix), o Apache também conta com versões para o Windows, para o Novell Netware e para o OS/2, o que o torna uma ótima opção para rodar em computadores obsoletos (desde que este atenda aos requisitos mínimos de hardware).

O servidor Apache é capaz de executar código em PHP, Perl, Shell Script e até em ASP e pode atuar como servidor FTP, HTTP, entre outros. Sua utilização mais conhecida é a que combina o Apache com a linguagem PHP e o banco de dados MySQL (combinação usada aqui no InfoWester).

A exigência de *hardware* do Apache depende de sua aplicação, mas um PC Pentium com 64 MB de memória RAM é capaz de executá-lo tranquilamente em um ambiente corporativo pequeno. No entanto, quando se trata de um *site* na internet, é interessante ter máquinas tão poderosas quanto o que exige o nível de acesso.

- Tem suporte a scripts cgi usando linguagens como Perl, PHP, Shell Script, ASP, etc;
- Suporte a autorização de acesso podendo ser especificadas restrições de acesso separadamente para cada endereço/arquivo/diretório com acesso no servidor;
- Autenticação requerendo um nome de usuário e senha válidos para acesso a alguma página/sub-diretório/arquivo (suportando criptografia via Crypto e MD5);
- Negociação de conteúdo, permitindo a exibição da página Web no idioma requisitado pelo Cliente Navegador;

- Suporte a tipos mime;
- Personalização de logs;
- Mensagens de erro;
- Suporte a virtual hosting (é possível servir 2 ou mais páginas com endereços/ portas diferentes através do mesmo processo ou usar mais de um processo para controlar mais de um endereço);
- Suporte a IP virtual *hosting*;
- Suporte a *name virtual hosting*;
- Suporte a servidor Proxy ftp e http, com limite de acesso, *caching* (todas flexivelmente configuráveis);
- Suporte a proxy e redireccionamentos baseados em URLs para endereços Internos;
- Suporte a criptografia via SSL, Certificados digitais;
- Módulos DSO (*Dynamic Shared Objects*) permitem adicionar/remover funcionalidades e recursos sem necessidade de recompilação do programa.

3.9 PHPStorm

JetBrains PhpStorm é um, cross-plataforma comercial IDE para PHP construído sobre JetBrains ' IDEA IntelliJ plataforma. PhpStorm fornece um editor para PHP, HTML e JavaScript com análise de código on-the-fly, prevenção de erros e automatizados refatorações para código PHP e JavaScript. Os usuários podem estender o IDE através da instalação de *plugins* criados para a Plataforma IntelliJ ou escrever seus próprios *plugins*.

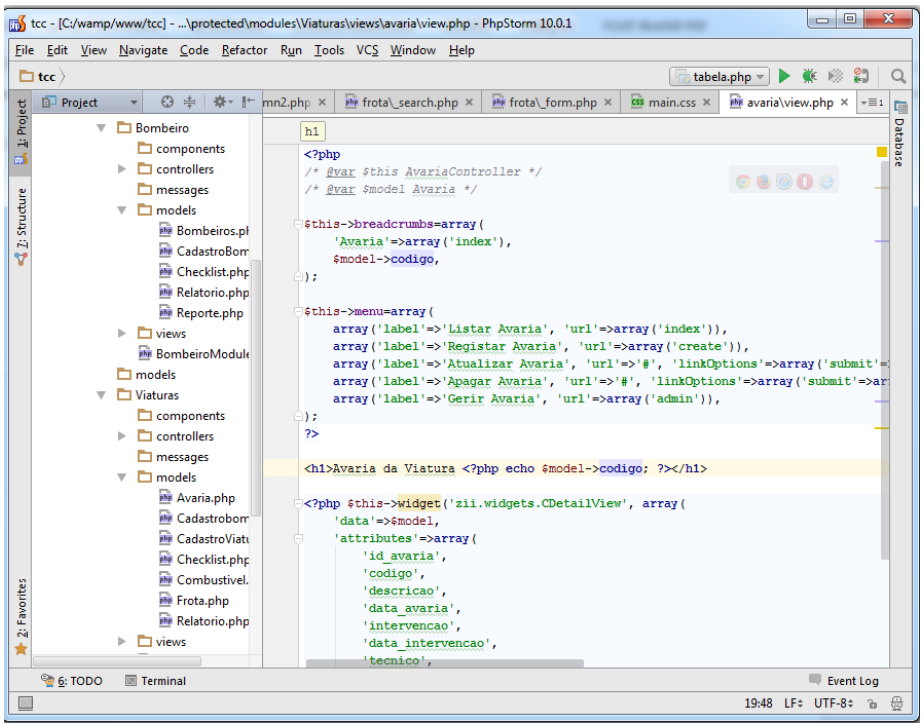


Figura 12 - Imagem do PHPStorm

Do PhpStorm conclusão de código suporta PHP 5.3, 5.4, 5.5, 5.6 e 7.0 (projetos modernos e de legado), incluindo geradores , co-rotinas , o finalmente, palavra-chave, lista no *foreach*, *namespaces* , fechamentos , traços e sintaxe de matriz curta. Inclui um pleno direito SQL editor com os resultados da consulta editáveis.

Todos os recursos disponíveis no **WebStorm** estão incluídos na PhpStorm, que adiciona suporte para PHP e banco de dados. Navios WebStorm com *plugins* pré-instalados JavaScript (como por Node.js), que estão disponíveis para PhpStorm bem, sem nenhum custo. (29)

3.10 UML

Linguagem de Modelagem Unificada (do inglês, **UML - Unified Modeling Language**) é uma linguagem de modelagem que permite representar um sistema de forma padronizada.

A UML não é uma metodologia de desenvolvimento, o que significa que ela não diz o que fazer primeiro e em seguida ou como projetar um sistema, mas ela auxilia a visualizar o desenho e a comunicação entre os objetos.

Basicamente, a UML permite que desenvolvedores visualizem os produtos de seus trabalhos em diagramas padronizados. Junto com uma notação gráfica, a UML também especifica significados, isto é, semântica. É uma notação independente de processos, embora o RUP (*Rational Unified Process*) tenha sido especificamente desenvolvido utilizando a UML.

É importante distinguir entre um modelo UML e um diagrama (ou conjunto de diagramas) de UML. O último é uma representação gráfica da informação do primeiro, mas o primeiro pode existir independentemente. O XMI (XML *Metadata Interchange*) na sua versão corrente disponibiliza troca de modelos mas não de diagramas.

Os objetivos da UML são: especificação, documentação, estruturação para sub-visualização e maior visualização lógica do desenvolvimento completo de um sistema de informação. (30)

4 CAPITULO IV (31)

4.1 Caracterização da ASA

4.1.1 A Empresa

A 17 de Fevereiro de 1984, foi criada a asa – E.P., Empresa Nacional de Aeroportos e Segurança Aérea, que passou a designar-se por asa – S.A., a partir de Junho de 2001, quando passou a Sociedade Anónima e a reger-se pelo código das empresas comerciais, mantendo-se, no entanto, a titularidade de todos os direitos e obrigações de que era detentora a asa – E.P.

Trinta e dois anos depois da sua criação e dos progressos conseguidos a vários níveis, a asa encontra-se numa fase importante do seu ciclo de vida que se caracteriza por uma firme e continuada aposta no desenvolvimento tecnológico, na modernização das infraestruturas e na capacitação dos recursos humanos, por forma a dotar-se de instrumentos de gestão compatíveis com os desafios da atualidade, permitindo-lhe a competitividade, o dinamismo para acompanhar as mudanças no sistema dos transportes aéreos a nível mundial.

4.1.2 Missão

A asa, S.A. tem por objeto principal a exploração e o desenvolvimento em moldes empresariais e em regime exclusivo do serviço público de apoio à aviação civil, a gestão do tráfego aéreo, garantindo os serviços de partida, sobrevoo e chegada de aeronaves, a gestão dos terminais de carga e correios, assegurando para isso as atividades e serviços inerentes às infraestruturas aeronáuticas e de navegação aérea, em todos os aeroportos e aeródromos públicos de Cabo Verde e na Região de Informação de Voo Oceânica do Sal, designada por FIR Oceânica do Sal.

4.2 Estrutura Organizacional

O modelo organizacional da ASA é composto por órgãos de diferentes natureza, com competências e atribuições distintas, nomeadamente:

I. Órgãos de assessoria;

II. Órgãos de suporte;

III. Órgãos operacionais.

Natureza dos Órgãos

Órgãos de assessoria: Garantem todo o apoio ao concelho de administração em termos de planeamento e execução das atividades e do desenvolvimento estratégico do negócio.

4.2.1 Exemplos de atividades:

- Planeamento estratégico
- Apoio Jurídico
- Segurança
- Qualidade
- Ambiente
- Comunicações e relações externas
- Controlo de gestão

Órgãos de suporte: Responsáveis pela coordenação e execução de atividades de suporte a toda a organização em termos de gestão administrativa, de recursos humanos, financeiros, informáticos e de manutenção.

Exemplos de atividades:

- Recursos Humanos
- Comercial
- Financeiro
- Informático
- Manutenção Geral

Órgãos Operacionais: Responsáveis pelo planeamento, coordenação e execução das atividades nucleares.

Exemplo de atividades:

- Navegação aérea
- Gestão aeroportuária

4.3 Investimentos

O *Business Plan* 2014 – 2018 estabelece um programa de investimentos orçado em cerca de 6.4 milhões de contos, e visa essencialmente consolidar os investimentos de capacidade iniciados no quinquénio anterior, que centrava principalmente na modernização e capacidade das infraestruturas aeroportuárias por forma a responder as demandas de crescimento de trafego.

5 CAPITULO V

5.1 Análise do Sistema

5.1.1 Especificação dos requisitos

Para que ficasse claros os requisitos do sistema, foram divididos e duas categorias:

- Requisitos funcionais

- Requisitos não funcionais

5.1.2 Requisitos Funcionais

Este reqisito define uma função de um sistema de software ou seu componente. Uma função é descrita como um conjunto de entradas, seu comportamento e as saídas. Para esta aplicação foram definidas os seguintes requisitos funcionais:

- **Tratamento dos Check-List** – O sistema tem um controlo de gestão de Check-List, desde preenchimento, atualização, listagem e gestão;
- **Gestão de Tarefas** – O sistema permite agendar tarefas com datas estipuladas que ficaram em aberto até resolução destas;
- **Gestão de Combustíveis** – O sistema permite o registo de todo o combustível gasto, listagem dos gastos e das viaturas reabastecidas;
- **Gestão dos Reportes –** O sistema permite o registo dos reportes que aconteçam no serviço e a respetiva listagem;
- **Gestão dos Extintores –** O sistema permite a gestão dos extintores desde o cadastro, identificação completa do equipamento, listagem dos extintores;
- **Tratamento de Relatório de Turnos –** O sistema permite a elaboração dos relatórios**,** listagem, atualização e correção;
- **Cadastro dos Bombeiros** – O sistema permite o cadastro dos Bombeiros e gestão da informação referente;
- **Cadastro das Viaturas** – O sistema permite o cadastro das viaturas, listagem e atualização caso necessário;
- **Gestão de Avarias** – O sistema permite o lançamento das avarias, listagem, e tratamento caso a resolução da avaria.

5.2 equisitos não Funcionais

- **Implementação** – O sistema foi desenvolvida na linguagem PHP, CSS, JQuery, HTML;

- **Servidor de Aplicação** – O sistema e o banco de dados é suportado pelo servidor ***apache;***
- **Bancos de Dados – Os dados são guardados** nuns bancos de dados elaborados no PHPmyAdmin, na linguagem SQL;
- **Portabilidade** – O sistema deverá ser instalada em um servidor e aberto através de um navegador desde que tenha internet;
- **Navegador** – O sistema pode ser aberto em qualquer navegador;
- **Facilidade** – O sistema é de fácil perceção sendo que é de linguagem como mesmo sendo uma aplicação pra uma área bastante técnica.
- **Escalabilidade** – O sistema tem uma carga de escalabilidade bastante alta, seja ela geográfica ou administrativa, sendo que vai estar alojada num servidor.

5.3 Modelação do Sistema

Modelagem de Sistemas é a atividade de construir modelos que expliquem as características ou o comportamento de um *software* ou de um sistema de *software*. Na construção do *software* os modelos podem ser usados na identificação das características e funcionalidades que o software deverá prover (análise de requisitos), e no planeamento da sua construção.

Frequentemente a modelagem de *software* usa algum tipo de notação gráfica e são apoiados pelo uso de Ferramentas CASE.

A modelagem de *software* normalmente implica a construção de modelos gráficos que simbolizam os artefactos dos componentes de software utilizados e os seus inter-relacionamentos. Uma forma comum de modelagem de programas procedurais (não orientados a objeto) é através de fluxogramas, enquanto a modelagem de programas orientados a objeto normalmente usam a linguagem gráfica UML. [(32)]

5.4 Diagrama de Casos de Uso

O **diagrama de caso de uso** descreve a funcionalidade proposta para um novo sistema que será projetado, é uma excelente ferramenta para o levantamento dos requisitos funcionais do sistema. Um caso de uso é um "documento narrativo que descreve a

sequência de eventos de um ator que usa um sistema para completar um processo" (Ivar Jacobson). Um caso de uso representa uma unidade discreta da interação entre um usuário (humano ou máquina) e o sistema. Um caso de uso é uma unidade de um trabalho significante. Por exemplo: o "*login* para o sistema", "registrar no sistema" e "criar pedidos" são todos casos de uso. Cada caso de uso tem uma descrição da funcionalidade que será construída no sistema proposto. Um caso de uso pode "usar" outra funcionalidade de caso de uso ou "estender" outro caso de uso com seu próprio comportamento. (33)

Casos de uso são tipicamente relacionados a "atores". Um ator é um humano ou entidade máquina que interage com o sistema para executar um significante trabalho.

Casos de uso são tipicamente relacionados a "atores". Um ator é um humano ou entidade máquina que interage com o sistema para executar um significante trabalho.

- **Administrador:** É o ator com mais privilégios, tem acesso a todos os perfis do sistema;
- **Chefe de Serviço:** É o ator que tem acesso a perfis específicos para monitorização do trabalho através do sistema;
- **Supervisor:** É o ator que tem acesso a perfis específicos, para elaboração de formulários, relatórios;
- **Bombeiros:** É o ator que tem acesso a perfis específicos, área operacional, para elaboração de formulários do trabalho no sistema.

5.5 Diagramas do Uso de Caso

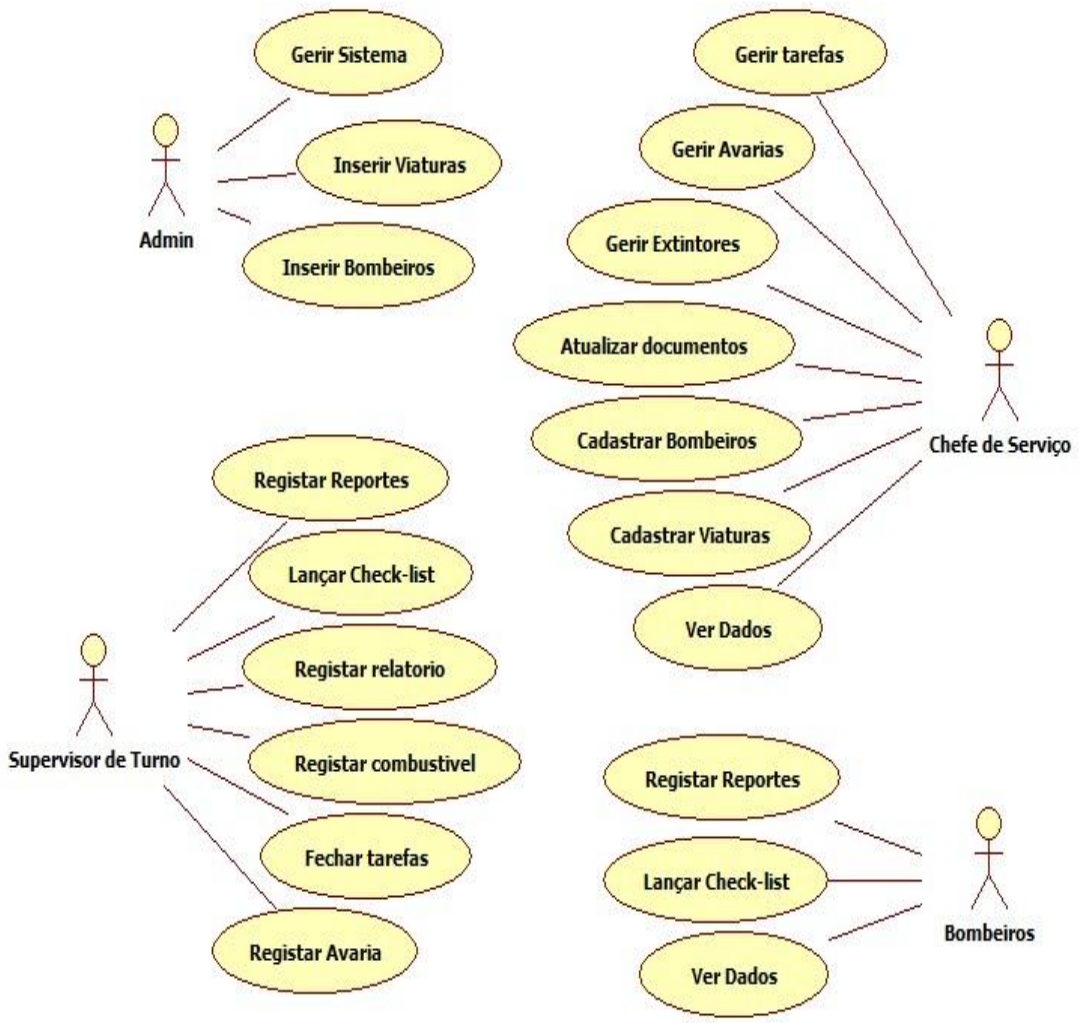


Figura 13 - Diagramas do Uso de Casos

5.6 Diagramas de Sequencia

5.6.1 Caso de Uso: Login

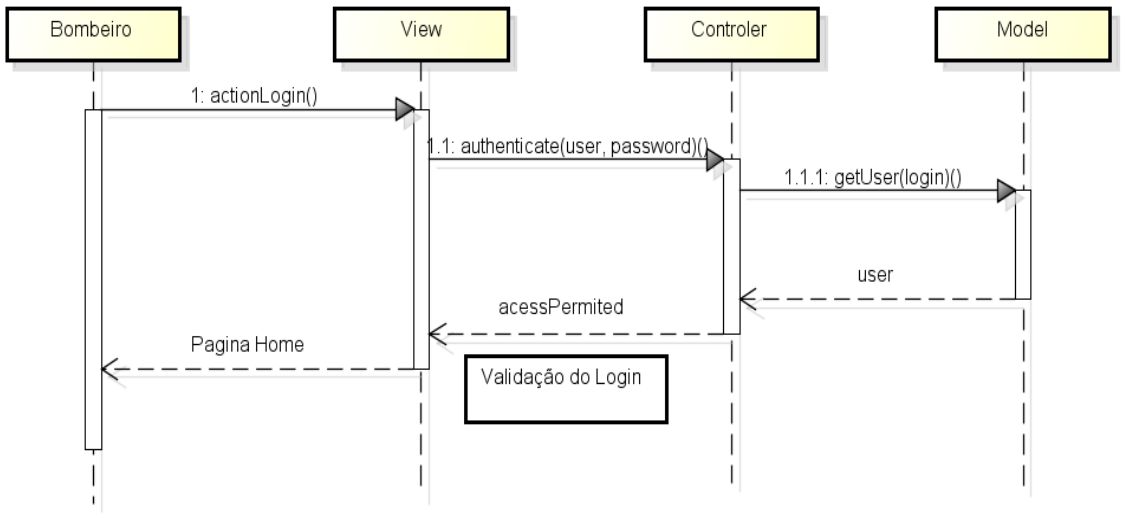


Figura 14 - Diagrama Sequencia: Login

5.6.2 Caso de Uso: Registar Extintor

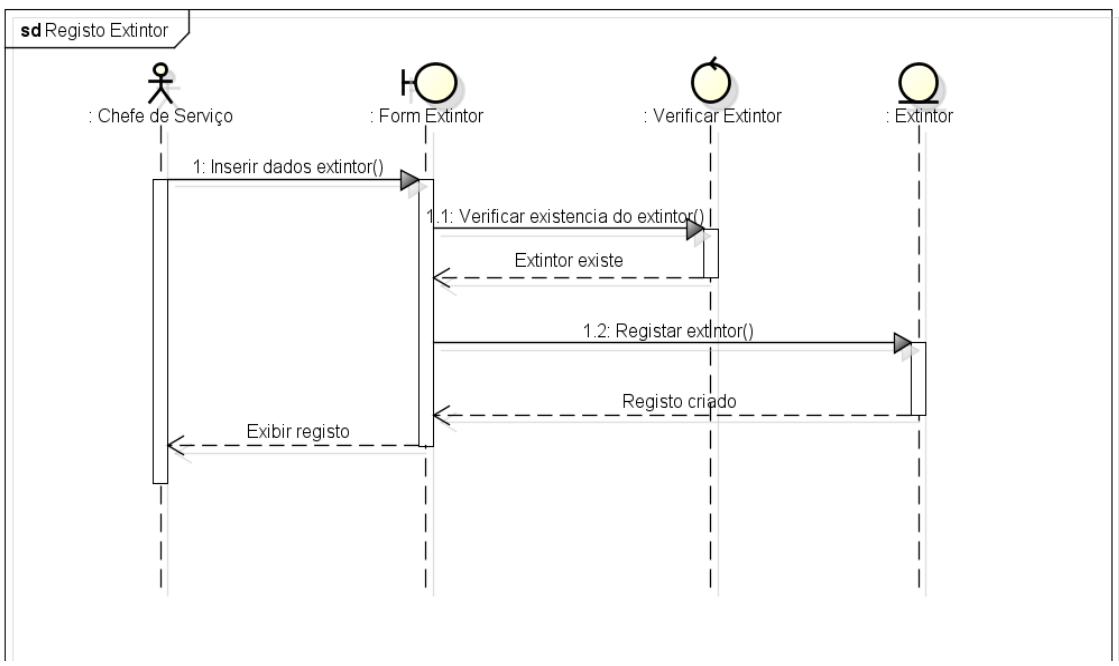


Figura 15 - Diagrama Sequencia: Registar extintores

5.6.3 Caso de Uso: Cadastrar Bombeiro

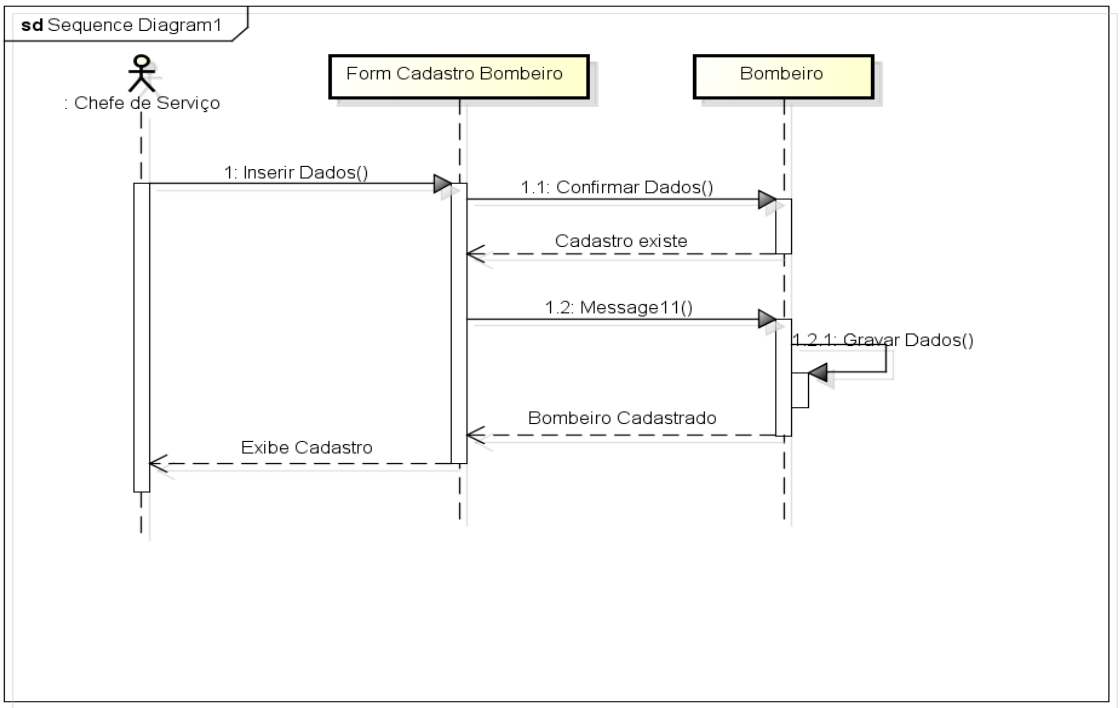


Figura 16 - Diagrama Sequencia: Cadastrar Bombeiro

5.7 Diagrama de Entidade e Relacionamento

O diagrama de entidade e relacionamento é um modelo abstrato cuja finalidade é descrever de maneira conceitual os dados a serem utilizados em um sistema de informação e pode ser representado por entidades, relacionamentos e atributos.
Existem muitas notações para diagrama de entidades e relacionamentos. A notação original proposta por Peter Chen é composta de entidades (retângulos), relacionamentos (losangos), atributos (elipses) e linhas de conexão (linhas) que indicam a cardinalidade de uma entidade em um relacionamento. (Londeix, 1995). [34]

Para definir uma ideia do diagrama do aplicativo GestFrota foram escolhidos algumas entidades no sentido de mostrar a funcionalidade do sistema.

As entidades, frota, bombeiro, tarefa, viatura, relatório, reporte, cadastrobombeiro, cadastroviatura, extintores, avaria mostram o relacionamento entre si conforme figura 17.

5.8 Diagrama de Entidade Relacional

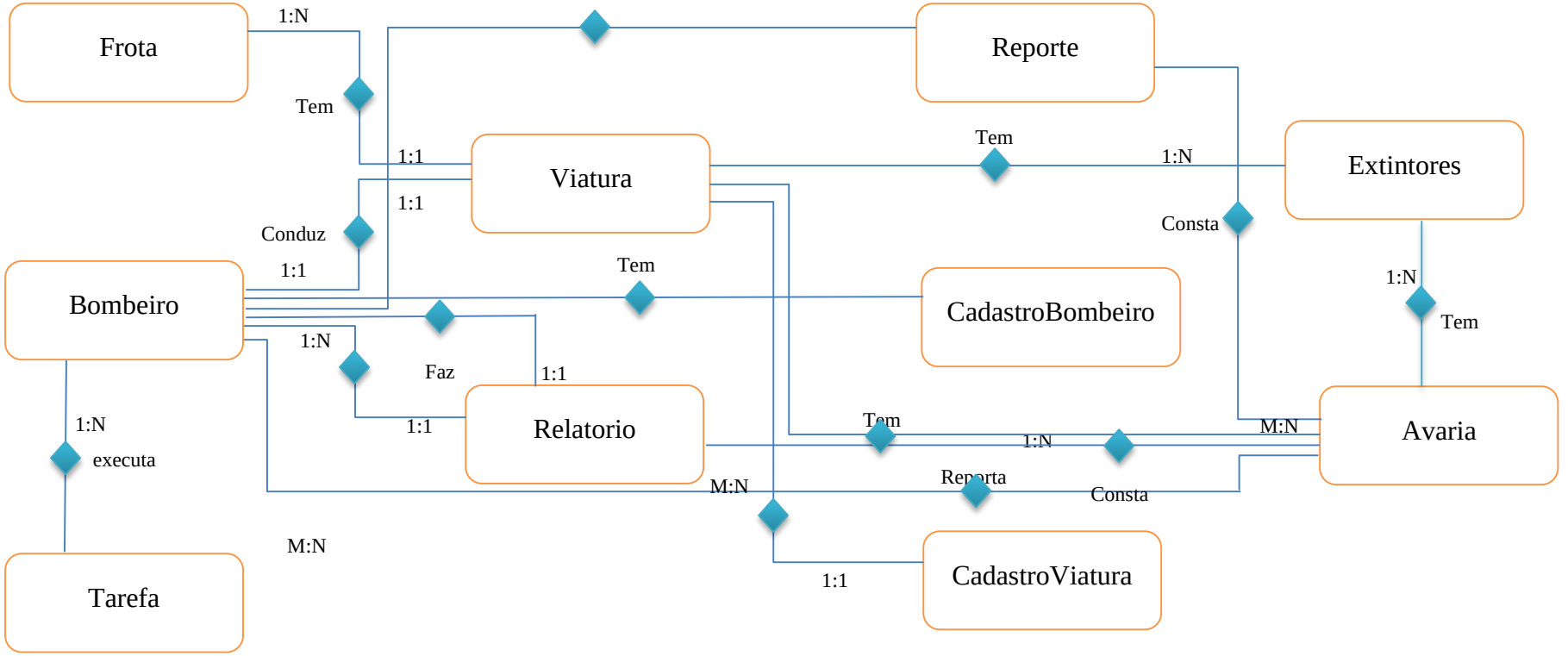


Figura 17 - Diagrama Entidade Relacional

6 CAPITULO VI

6.1 Protótipo do Sistema

Este capítulo vai mostrar as principais funcionalidades do protótipo do sistema proposto no trabalho, em que divide em quatro níveis de usuários.

- Chefe de Serviço (CSOSS) – este nível na empresa é direcionado aos chefes de serviço com o intuito de monitorizar todo o trabalho lançado no sistema, bem como alteração de formulários específicos na sua responsabilidade;
- Supervisor – nível em que faz o lançamento dos combustíveis, o preenchimento do relatórios, e alteração de qualquer formulário (Check-List) quando solicitado por um colaborador;
- Bombeiro – este nível não tem muitas responsabilidades no sistema. A tarefa será o preenchimento do Check-List das viaturas e o lançamento de reportes.

6.2 Descrição das funcionalidades do sistema

6.2.1 Telas principais

O sistema vai estar assento num servidor da empresa, no sistema intranet, será chamado através de um endereço www.gestfrota.asa.cv e será aberto a página principal (Home) que tem algumas informações do serviço sem que o usuário esteja cadastrado.

Juntamente com a página principal (Home) esta a pagina login para que os usuários tenham acesso ao sistema conforme nível atribuído.

Na pagina principal esta o calendário de exercícios de mesa a serem realizados no serviço de Bombeiros ao longo do ano, um calendário, informações uteis (sintéticos) sobre viaturas, equipamentos e manutenção, bem como procedimentos para voos de categoria 5 e 7.

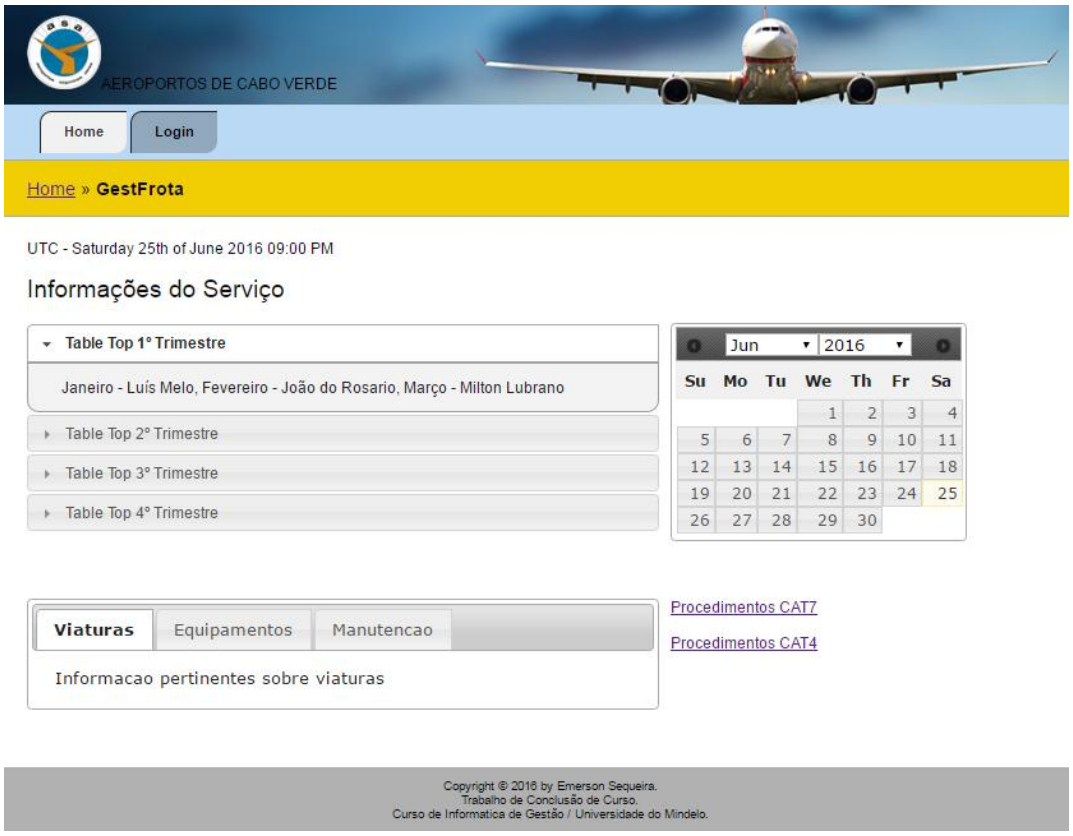


Figura 18 – Pagina principal do Sistema

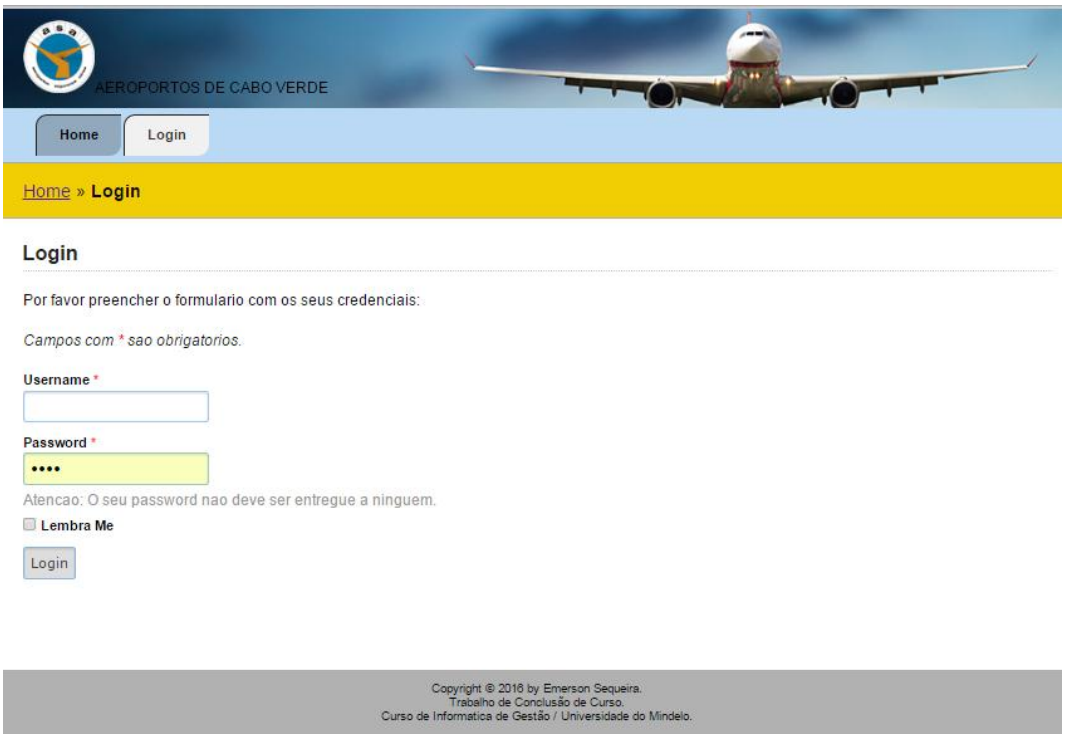


Figura 19 - Pagina Login do Sistema

6.3 Dentro do sistema

Ao entrar no sistema deparamos com um *layout* distribuído em 10 botões acionados através de ícones, que nos levara as funcionalidades dos sistema e sua operacionalidade, para o preenchimento dos formulários, consulta de dados, tratamento de informação da gestão do quartel no geral.

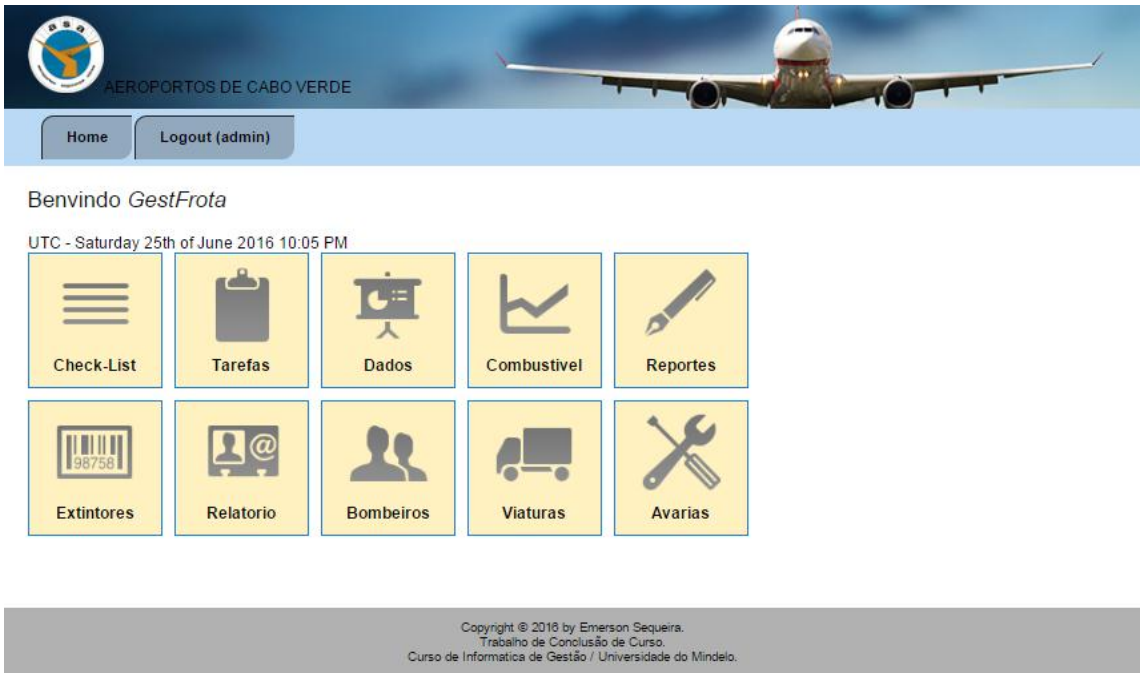


Figura 20 - Layout do Sistema após entrada

6.3.1 Check-List

Esta funcionalidade tem como objetivo o preenchimento dos Check-List das viaturas registadas no sistema, que aparecem através do *label* **viatura** *DropdownList* dinâmico.

Os Check-List são preenchidos pelos Bombeiros ou pelo supervisor, mas só podem ser alterados pelo supervisor ou o chefe de serviço evitando adulteração de informação.

No lado direito da tela está o menu da página onde podemos listar os Check-list anteriormente registados bem como a gestão dos mesmos.

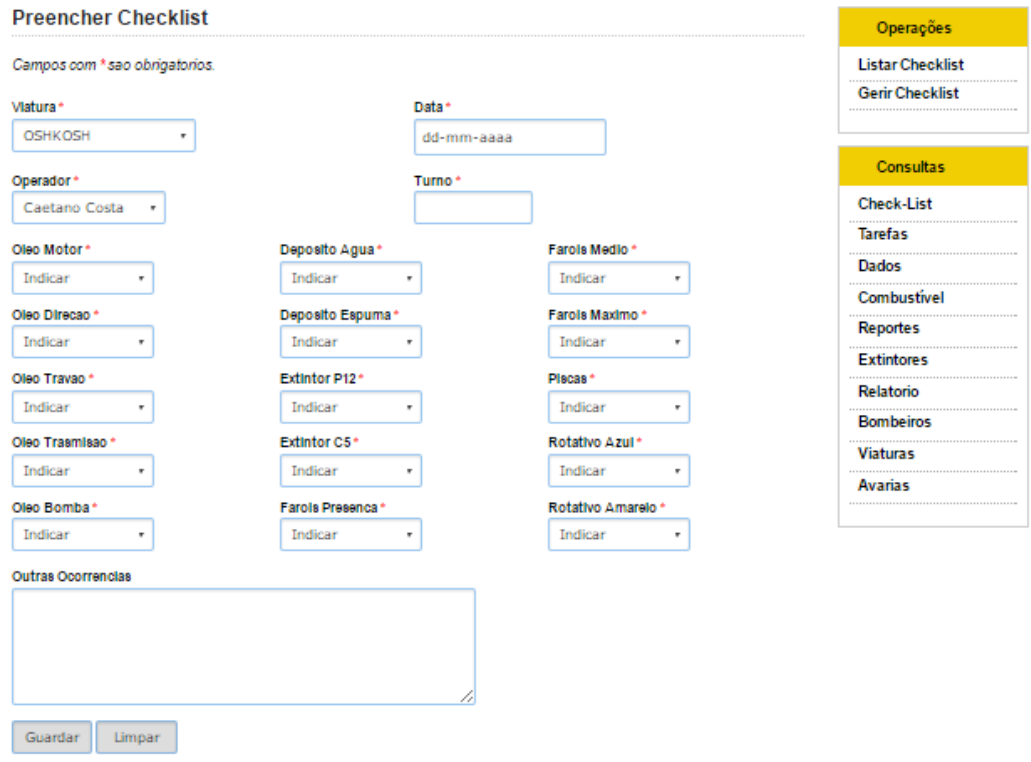


Figura 21 - Formulário Check-List

Para que o *dropDownList* Viatura funcionasse, foi preciso usar o comando findAll() conforme figura 22. O código percorre toda a tabela do banco de dados viatura, através da relação entre tabelas em que o código é a chave estrangeira da tabela **checklist**.

```
<div class="span-9">
    <div class="row">
        <?php echo $form->labelEx($model,'viatura'); ?>
        <?php echo $form->dropDownList($model,'viatura',CHtml::listData(frota::model()->findAll(),'codigo','marca')); ?>
        <?php echo $form->error($model,'viatura'); ?>
    </div>
    </div>
```

Figura 22 - Linha de código dinâmico

```
<div class="span-6">
    <div class="row">
        <?php echo $form->labelEx($model,'oleo_motor'); ?>
        <?php echo $form->dropDownList($model,'oleo_motor', array('Satisfacao'=>array(
                        'nulo'=>'Indicar',
                        'satisfaz'=>'Satisfaz',
                        'nao_satisfaz'=>'Nao Satisfaz',
                ),
        )); ?>
    </div>
```

Figura 23 - Linha de código de dropDownList estático

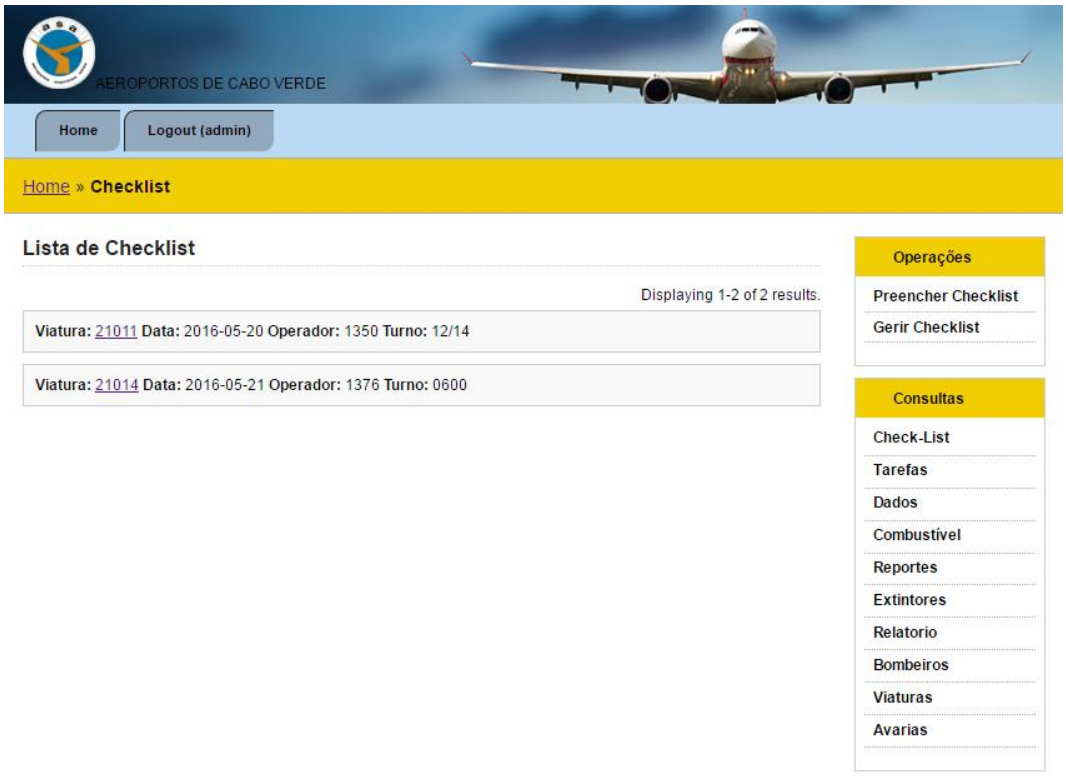


Figura 24 - Tela Listar Check-List

Na tela listar Check-list caso quisermos abrir um dos documentos já registados, basta clicar no código da viatura, sublinha a azul, para abrir o Check-List desejado.

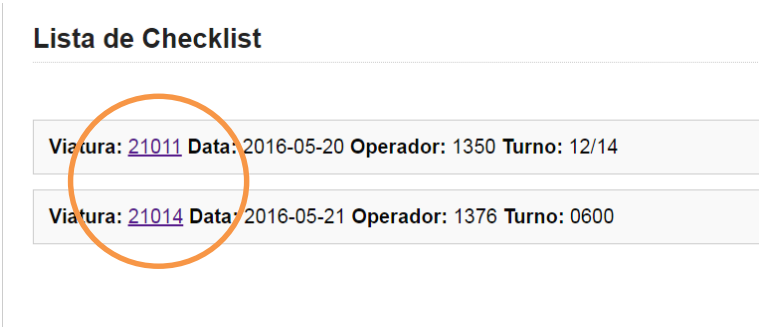


Figura 25 - Código de viatura (botão de abertura de Check-List)

Após a abertura do Check-list teremos todas as informações da viatura e do colaborador que a preencheu. O colaborador é identificado através do número mecanográfico da empresa.

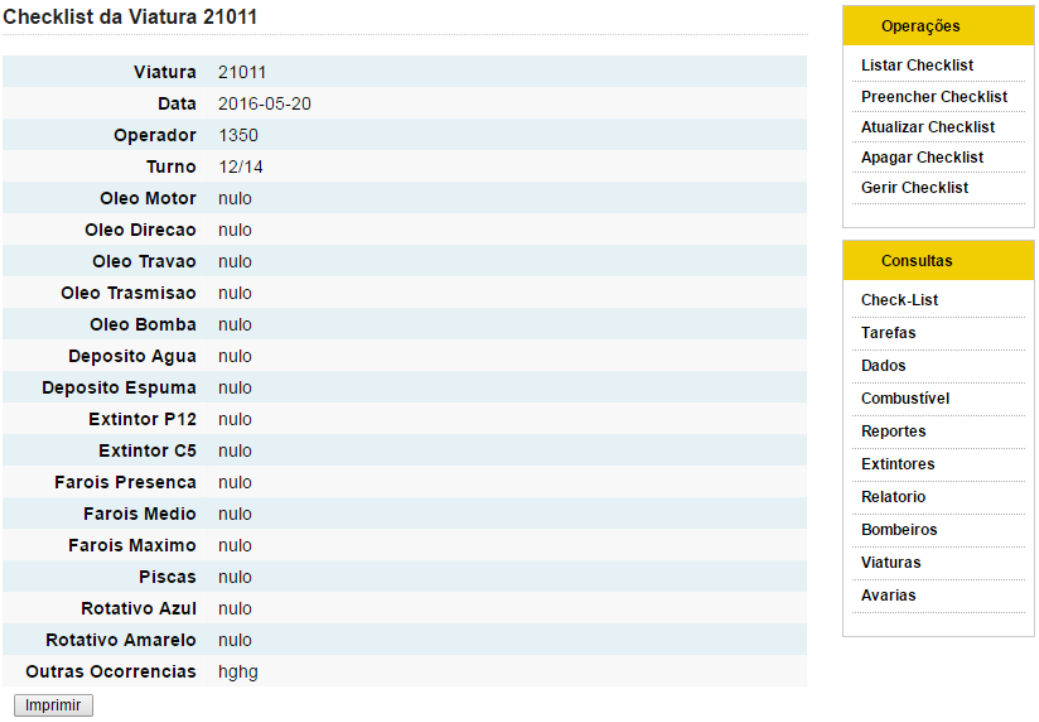


Figura 26 - Check-list da uma viatura após preenchimento

Para procurar um Check-List seja ela pelo colaborador que a preenche ou pelo código da viatura, basta acionar **Gerir Check-List** e teremos acesso a tela (figura 25) onde podemos usar os filtros para procura informação ou através da procura avançada.

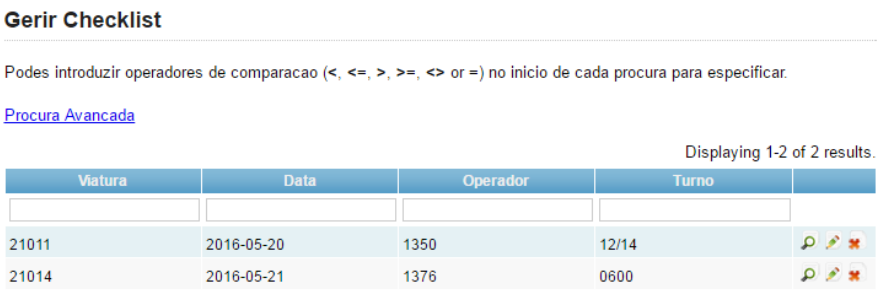


Figura 27 - Tela Gerir Check-List (Para procura de informação através de filtros)

Para apagar e abrir novo formulário Check-list, basta acionar o botão do primeiro menu do lado direito para a que a função desejada seja acionada.

6.3.2 Tarefas

A tela Tarefas é a onde o chefe de serviço emite uma tarefa para que seja executado por um supervisor ou equipa de trabalho. Ao acionar-mos o botão Tarefa abre todas as tarefas listadas, ordenadas a partir do mais recente.

Para abrir a tarefa basta acionar o nome da tarefa, nome em azul e sublinhado, para termos acesso a informação, conforme figura 26.

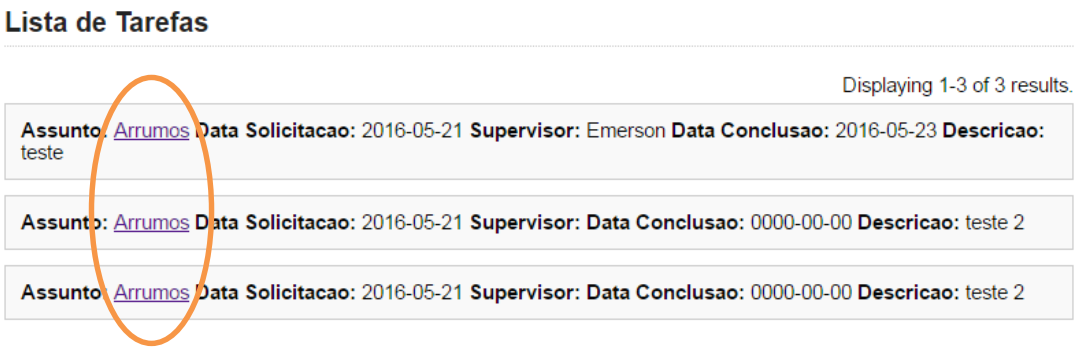


Figura 28 - Tela Lista de Tarefas

As tarefas ao serem lançadas estarão com a **Data Conclusão** do trabalho para que o recetor da informação dê o seu aval ao terminar a tarefa, assim a referida tarefa será considerada fechada.

6.3.3 Dados

A tela **Dados** é exclusivamente para a apresentação dos dados do serviço. Esta tela poderá ser atribuída a qualquer ao Diretor da empresa para que siga todas as informações em tempo real. Os dados são apresentados em gráficos conforme figura 27.

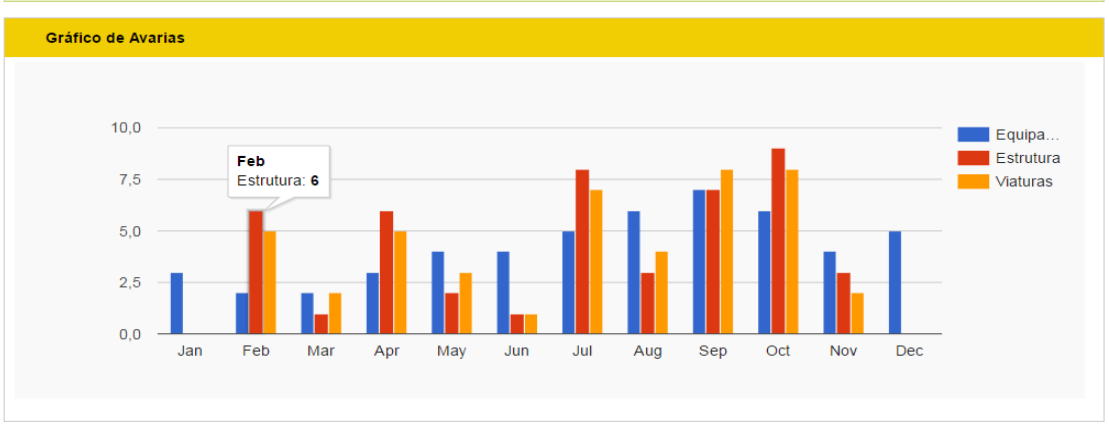


Figura 29 - Gráfico das avarias do serviço (Tela Dados)

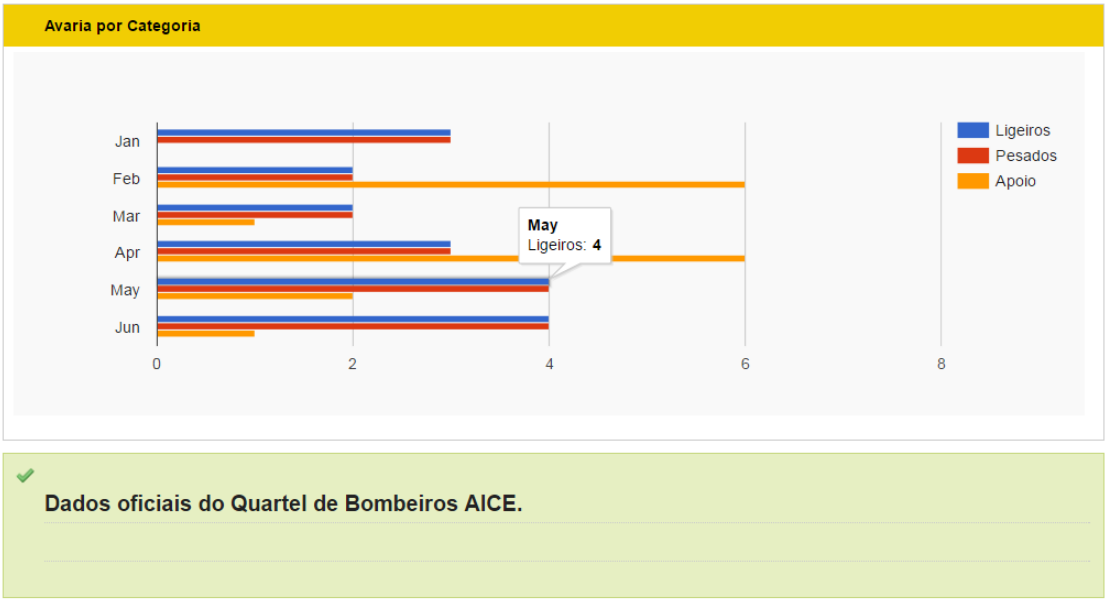


Figura 30 - Gráfico de avarias por categoria (Tela Dados)

6.3.4 Combustível

A tela de combustível é um formulário de registo de combustível para que todo o reabastecimento de combustível feito nas viaturas seja registado no banco de dados.

No formulário combustível tem um *DropDownList* dinâmico em que todas as viaturas registadas no sistema aparecem no formulário.

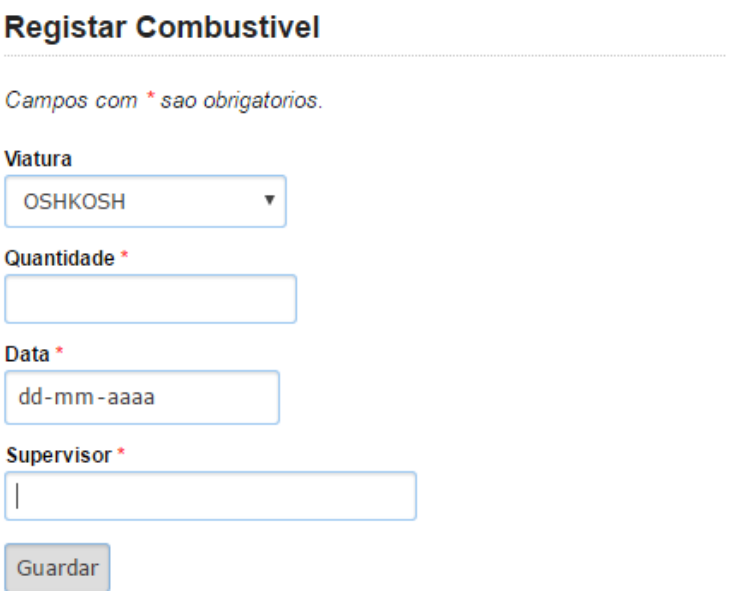


Figura 31 - Formulário Combustível

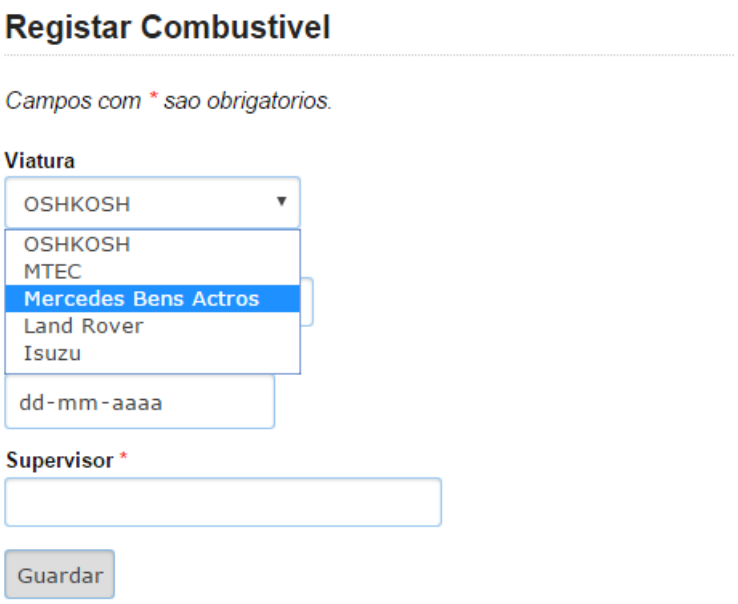


Figura 32 - Formulário Combustível (com *DropDownList* dinâmico aberto)

Na listagem de Combustível podemos ter acesso a todo o reabastecimento feito no serviço. Para abrir basta clicar no código da viatura e teremos acesso aos dados reabastecidos e o colaborador que a reabasteceu (ver figura 31).

Figura 33 - Tela Registo de Combustível

Caso quisermos informação sobre qualquer reabastecimento podemos ter acesso a informação através do menu – **Gerir Combustível** e procurar através do código da viatura, data.

6.3.5 Reporte

Esta tela tem permite os colaboradores fazerem o reporte de qualquer anomalia, acidente, incidente o até mesmo proposta sobre qualquer alteração necessária no serviço.

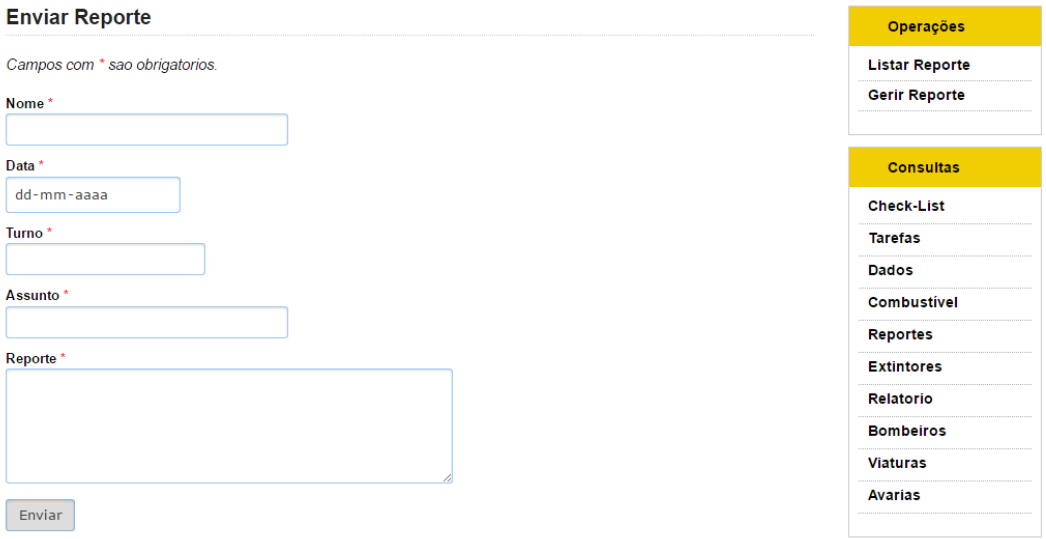


Figura 34 – Formulário Reporte

Para listar os reportes, basta recorrermos ao menu a direita da tela. O acesso leva-nos a tela da figura 33. Para abrir o reporte basta clicar no nome do emissor, em azul e sublinhado.

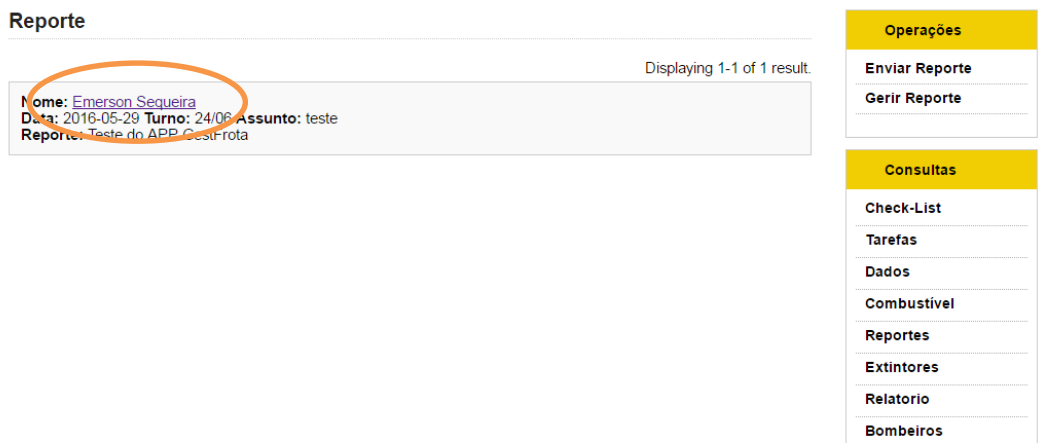


Figura 35 - Lista do Reportes registados no Banco de Dados

6.3.6 Extintor

Ao acionar-mos o botão da tela extintor, temos acesso lista de extintores registados no sistema. Todos os extintores são atribuídos um código (ex: 12003) para melhor identificação e registo único no sistema.

Para abri um registo de um extintor basta clicar na matrícula do extintor (figura 34) para temor acesso a todas as informações do extintor pretendido.

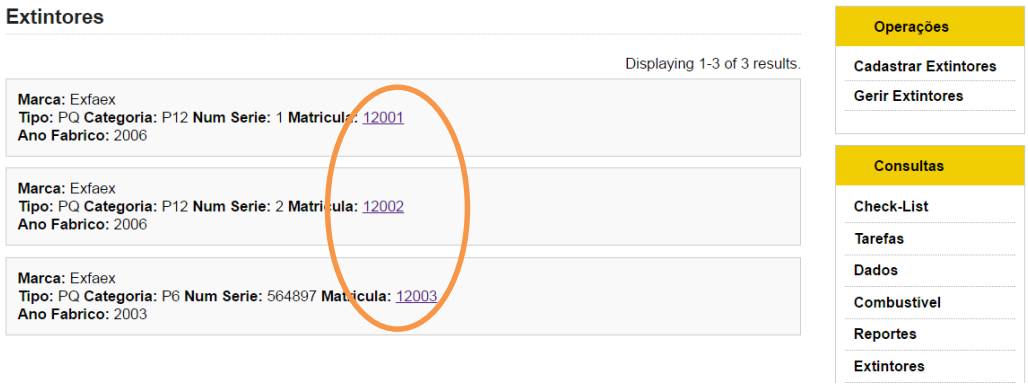


Figura 36 - Tela lista de todos os extintores registados no banco de dados

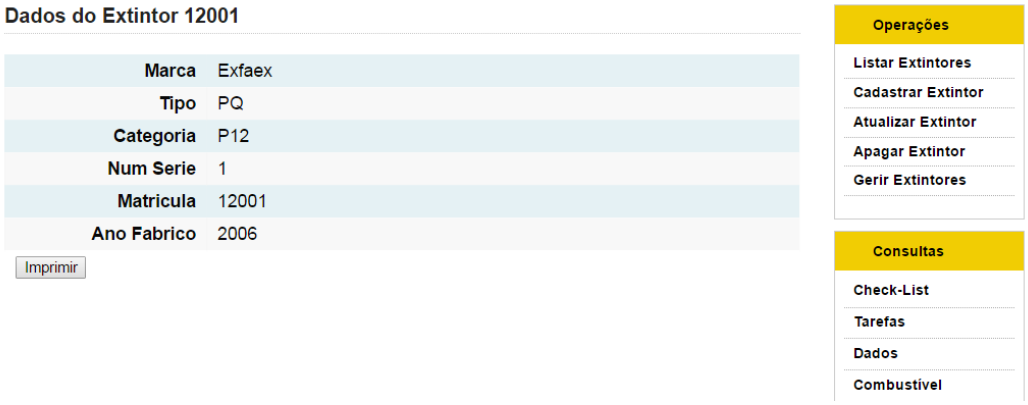
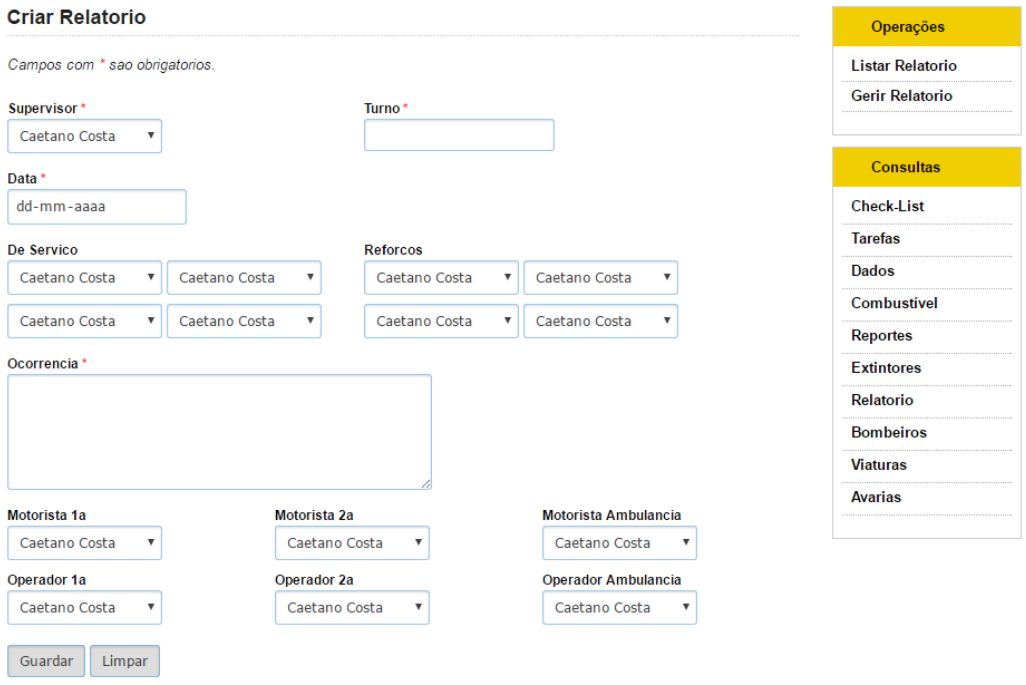


Figura 37 - Dados de um extintor (de matricula 12001)

6.3.7 Relatório



6.3.8 Bombeiros

Ao entrarmos no submódulo Bombeiros, somo confrontados com a lista de cadastros de todos os Bombeiros do serviço, figura 38;

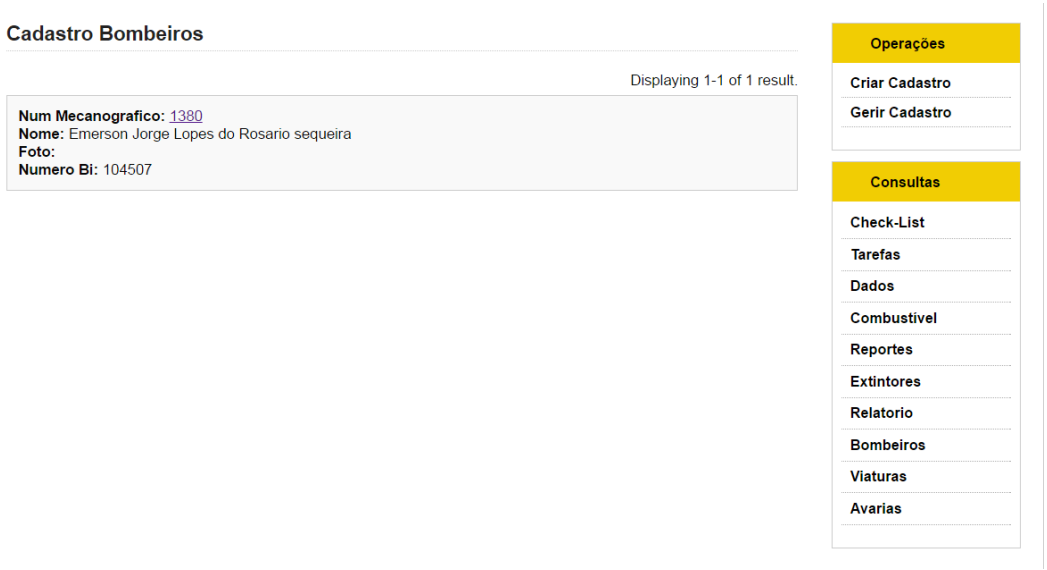


Figura 38 - Tela Lista Cadastro Bombeiros

Para termos acesso aos dados do cadastrado basta acionarmos o número mecanográfico.

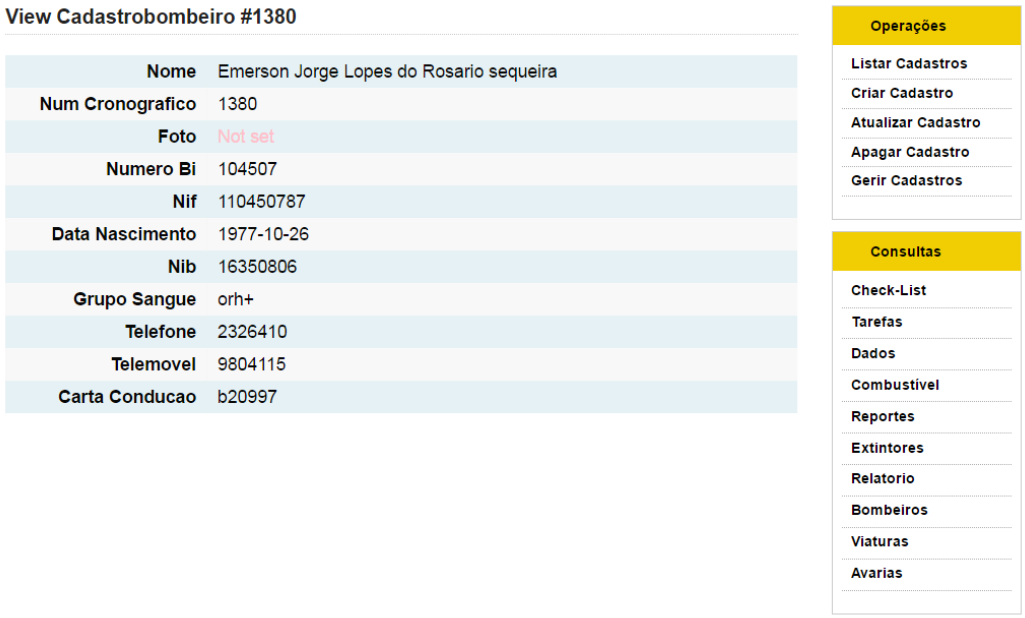


Figura 39 - Tela dos dados do Bombeiro Cadastrado

Caso quisermos procurar o cadastro de um colaborador, temos que abri no Menu **Gerir Cadastro,** abre a tela com os filtros de procura, figura 40.

Figura 40 - Tela dos filtros de procura de dados

6.3.9 Viaturas

Para o cadastro das viaturas temos o acesso através o botão Viaturas no início, que nos dá acesso ao formulário Cadastrar Viaturas, figura 41.

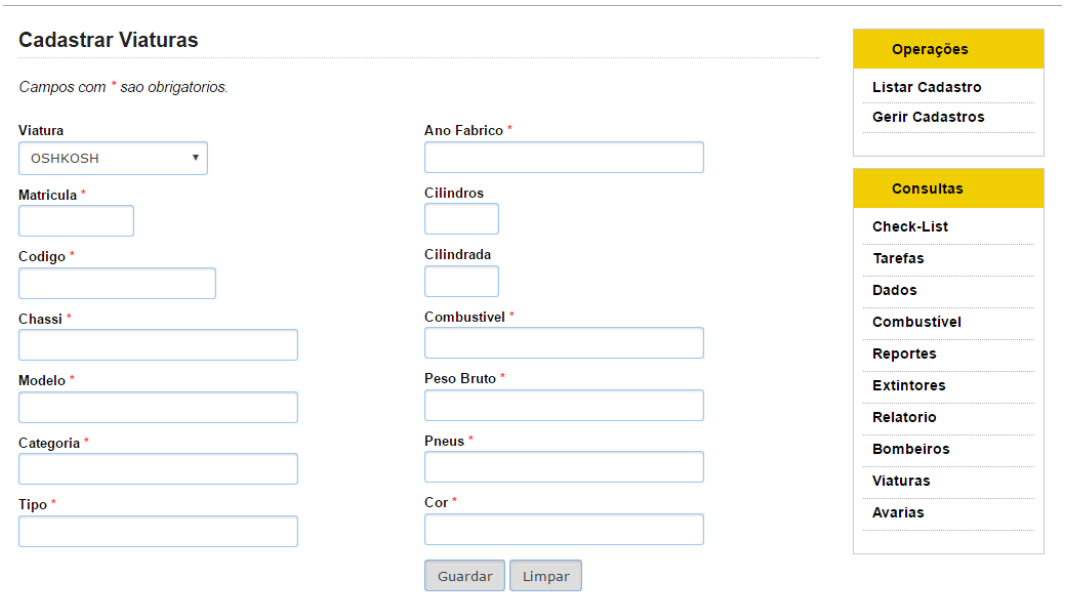


Figura 41 - Formulário de Cadastro de Viaturas

No formulário de cadastro temos um *dropDownList* dinâmico, o *dropDown* Viaturas, que faz a leitura de todas as viaturas que estejam registadas no sistema.

Ao cadastra-mos uma viatura é aberta todos os dados cadastrados com forma da confirmar-mos se os dados estão corretos, conforme figura 42.

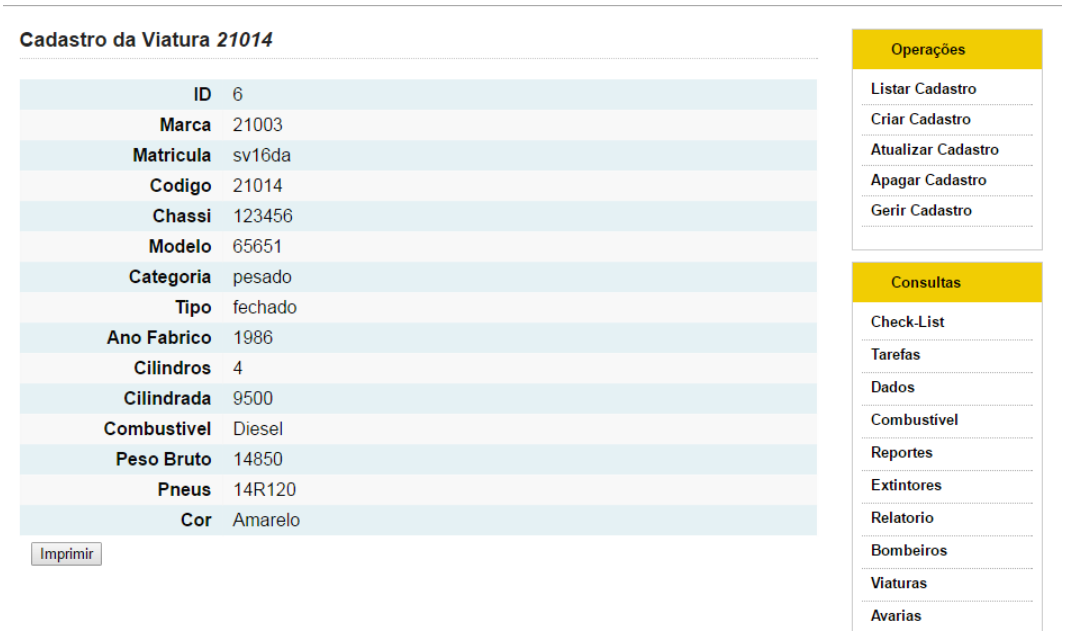


Figura 42 - Tela de dados cadastrados após validação

Após a confirmação dos dados caso quisermos alterar alguma informação basta clicar no Menu no lado direito, botão atualizar, e temos uma mensagem de confirmação de atualização. Ao confirmar a mensagem de atualização teremos o acesso novamente ao formulário com os dados a atualizar.

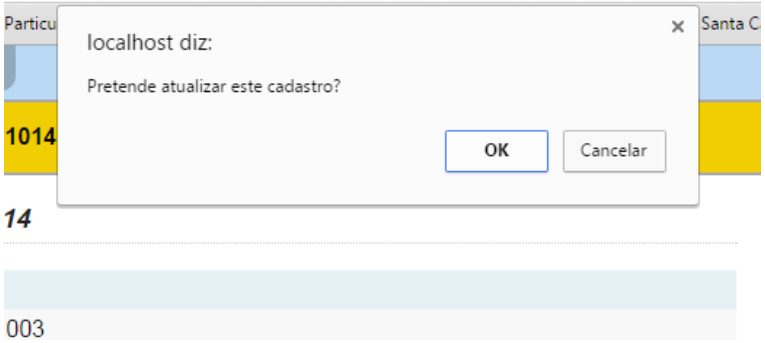


Figura 43 - Mensagem de confirmação de atualização de cadastro

6.3.10 Avarias

Este submódulo é para o registo das avarias que possam aparecer no serviço nas viaturas. Neste formulário não foi colocado nenhum *dropDownList* para escolha de viaturas sendo que a viatura é identificada pelo código atribuído pelo aeroporto.

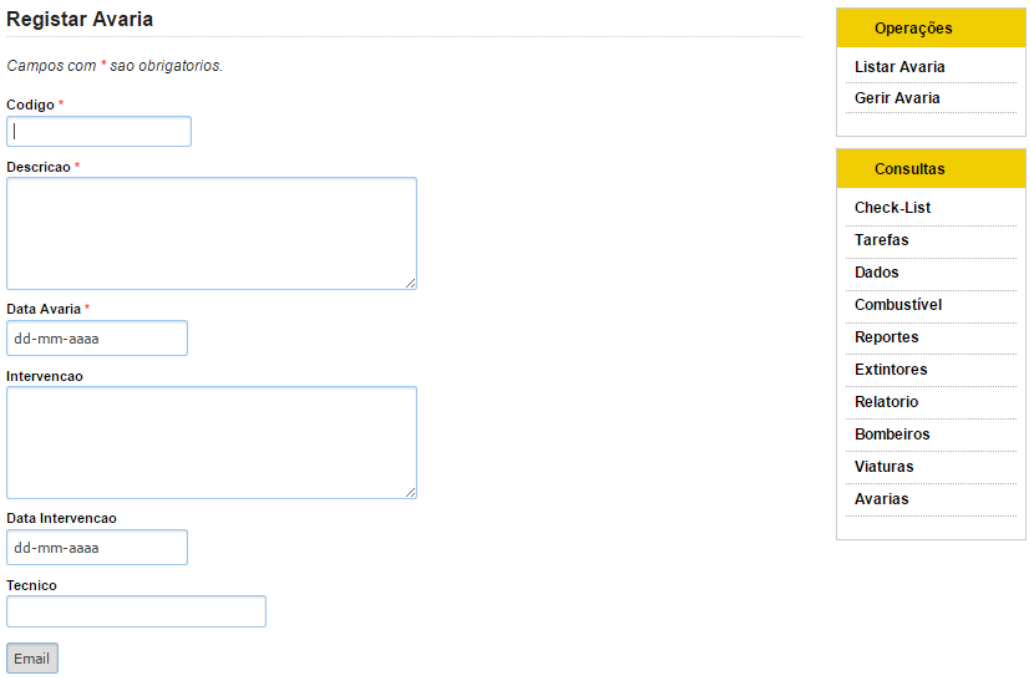


Figura 44 - Formulário de registo de avarias

Uma avaria ao ser registada no sistema, os atributos, **Intervencao**, **Data Intervencao** e **Tecnico**, ficam sem preencher até que a avaria seja resolvida. Neste caso consideramos a avaria em aberto e só é considerada fechada quando a viatura for reparada.

Figura 45 - Tela de avaria registada em estado aberto

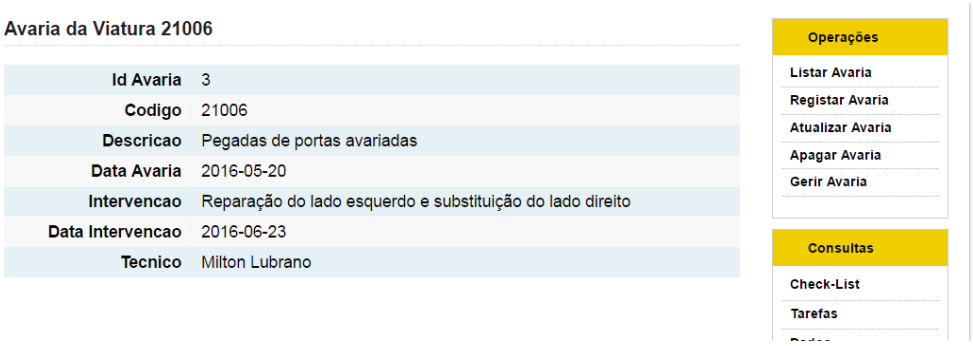


Figura 46 - Tela de avaria registada em estado fechado

6.4 Informações adicionais e Códigos usados no sistema

6.4.1 Como gerar códigos no Yii

Para criação de módulos e modelos no Yii, temos que chamar o gii para que possamos ter acesso ao gerador de código.

6.4.1.1 Ativação do gii:

- Abrir a pasta de instalação **www**;
- Entrar na pasta do protótipo;
- Abrir a pasta **protected**;
- Dentro da pasta **protected** encontramos o **config**;
- Entrar na pasta **config**;
- Abrir o ficheiro **main.php**.

Ao abrir-mos o ficheiro **main.php,** encontramos o código do yii comentado e para aciona-lo basta retira-lo da posição comentado (ver código abaixo).

```
'modules'=>array(
    // uncomment the following to enable the Gii tool
    'gii'=>array(
        'class'=>'system.gii.GiiModule',
        'password'=>'demo',
        // If removed, Gii defaults to localhost only. Edit carefully to
taste.
        'ipFilters'=>array('127.0.0.1','::1'),
```

```
    ),
    'Viaturas', 'Bombeiro',
),
```

Com a ativação do gii o Yii esta apto para gerar códigos. Para isso basta chamar o gii através do endereço **http://localhost/tcc/index.php?r=gii** e abre a janela de acesso como na figura 38.

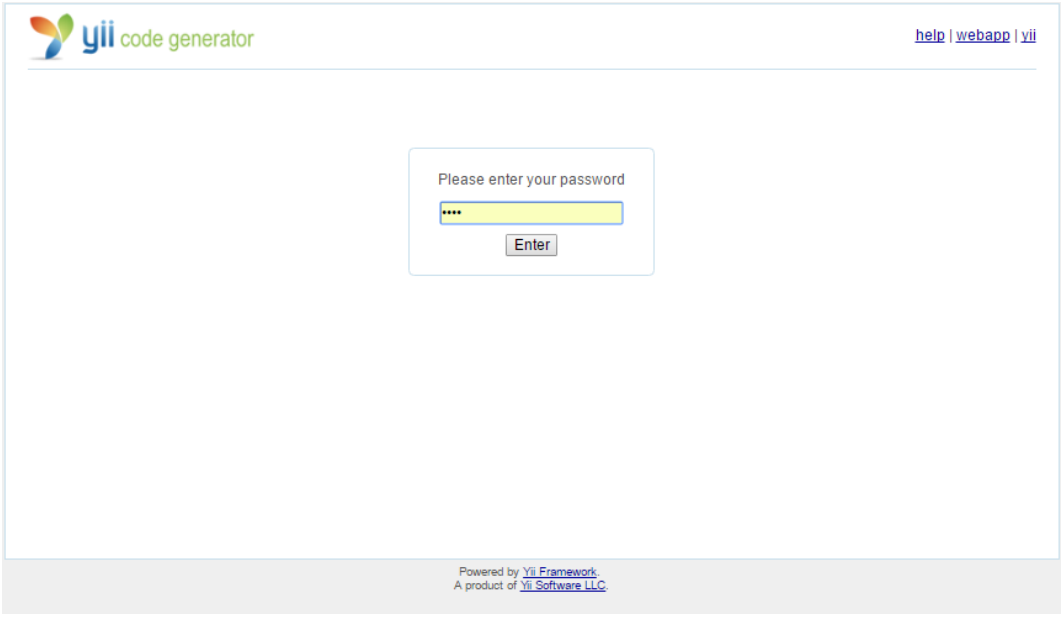


Figura 47 - Tela de acesso ao gerador de código Yii

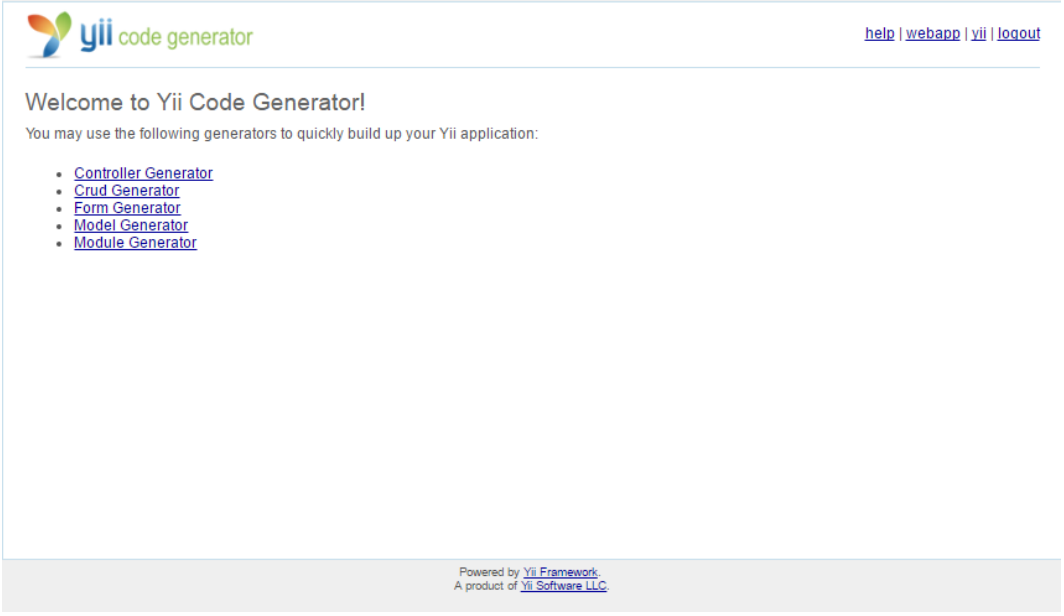
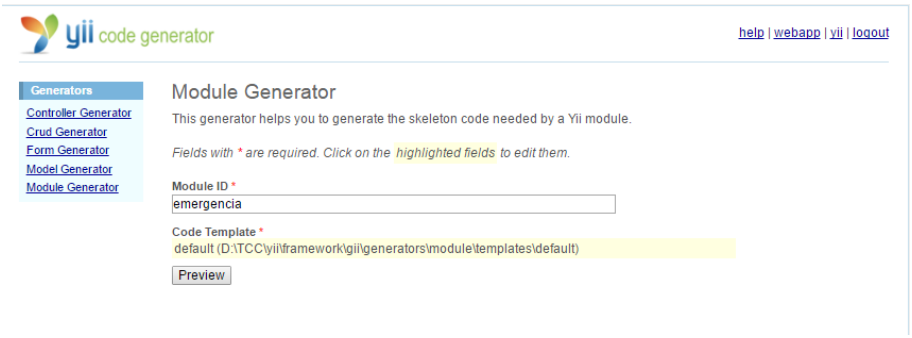


Figura 48 - Tela do gerador de código do Yii após acesso

6.4.2 Criar um módulo

- Clicar no botão module;
- Abre a janela Module ID, e inserimos o nome do módulo que queremos criar;



- Clicar em Preview após colocar o nome;
- É apresentado uma previsão dos códigos a serem gerados e aparecem na tela verde conforme figura 40;

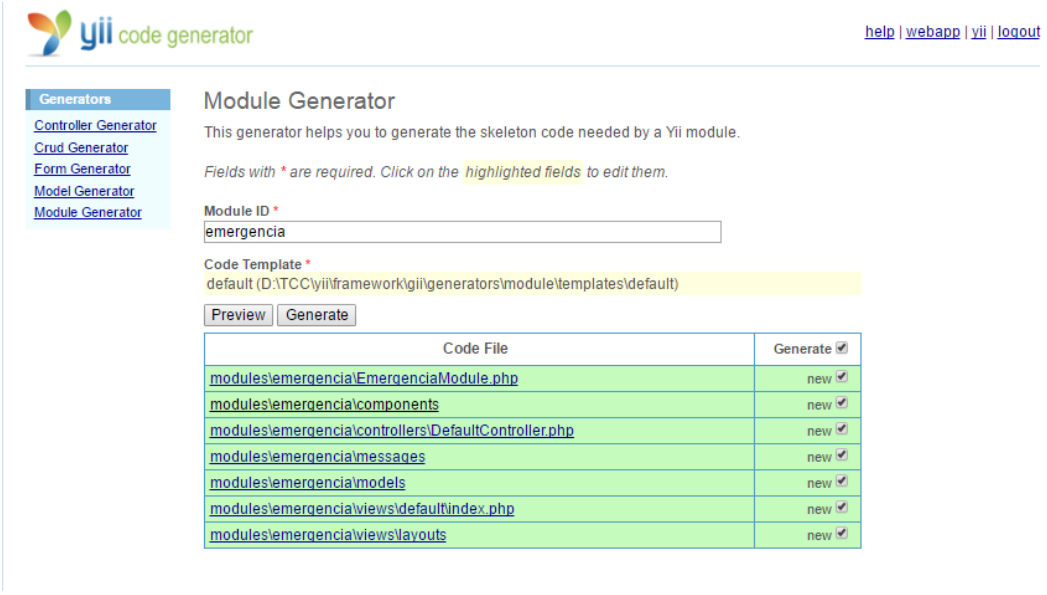


Figura 49 – Pré-visualização dos códigos a serem gerados para criação de um módulo

- Para gerar os códigos basta clicar em **Generate.**
- E temos a confirmação da criação dos códigos conforme figura 41.

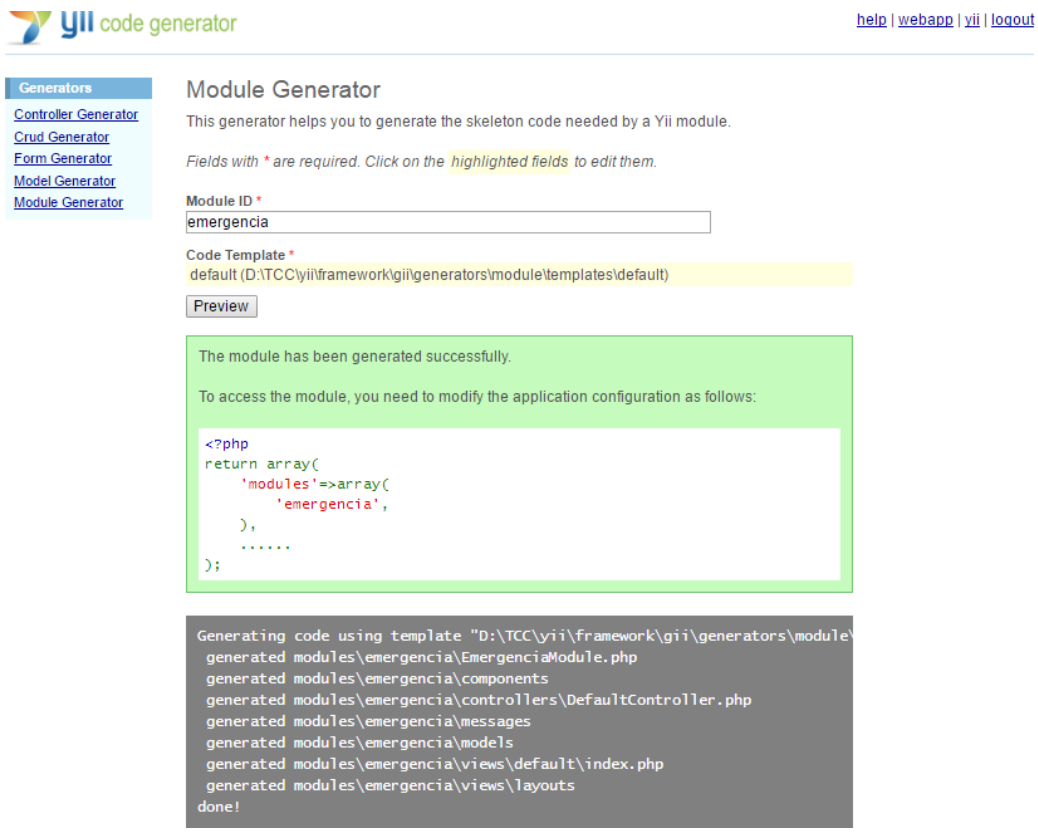


Figura 50 - Tela de confirmação dos códigos do módulo gerado

No sistema foi criado algumas regras para facilitar o tratamento de informação.

Foi criado um segundo Menu, no lado direito em baixo, para facilitar o acesso aos dados e todas as informações de listagem.

Figura 51 - Menu de consulta de dados listados

Por questões de segurança e evitar o apagar de dados por acidente foi implementado regras de confirmação nos botões de atualizar e apagar, através do comando *confirm* e este deve estar dentro de um vetor.

```
<div class="row buttons">
    <?php echo CHtml::submitButton($model->isNewRecord ? ' Guardar ' : ' Guardar ', array('confirm'=>'Pretende guardar o formulario?')); ?>
    <?php echo CHtml::resetButton($model->isNewRecord ? '  Limpar  ' : ' Limpar ',array('confirm'=>'Pretende limpar o formulario?')); ?>
</div>
```

Figura 52 - Linha de código de confirmação de Apagar e Atualizar

6.4.3 Código de acesso ao sistema dos diferentes níveis de usuários

```
public function accessRules()
{
    return array(
        array('allow',  // permite todos os usuarios 'index' e 'view' 
actions
            'actions'=>array('index','view'),
            'users'=>array('*'),
        ),
        array('allow', // permite o usuario 'create' e 'update' actions
            'actions'=>array('create','update'),
            'users'=>array('@'),
        ),
        array('allow', // permite o usuario admin 'admin' e 'delete' 
actions
            'actions'=>array('admin','delete'),
            'users'=>array('admin'),
        ),
        array('deny',  // deny all users
            'users'=>array('*'),
        ),
    );
}
```

7 Capitulo VII

7.1 Avaliação de metodologia

O sucesso de qualquer sistema depende do trabalho prévio realizado na escolha das aplicações para seu desenvolvimento.

Este trabalho esteve adaptado nas caraterística das ferramentas utilizadas, sendo mais-valia no rápido desenvolvimento e agilidade na criação do sistema. Contudo as escolhas

das ferramentas acabaram por ser um aspeto fundamental para o resultado final do sistema.

7.2 Conhecimentos adquiridos

O desenvolvimento deste sistema foi uma das maiores experiências já tida por mim neste percurso universitário. Todo o conhecimento em várias situações me obrigou a uma interação com vários intervenientes de outros projetos, uma grande aproximação em termos técnicos cm o orientador do projeto Eng.º Ciríaco Brito no aprofundar de temas de grande importância na elaboração deste projeto. Que de certeza serão aplicadas no futuro em outros projetos bem como no desenvolvimento mais alargado deste.

O uso do framework yii e da sua capacidade de desenvolvimento, bem como a sua segurança, dão maior enfase ao trabalho.

Aprofundar os conhecimentos em PHP orientado a objetos e sua vantagem na programação das aplicações web, é notório em geral.

Desenvolvimento na manipulação de base de dados, precisamente SQL.

7.3 Conclusão

Este trabalho foi abordado sobre o desenvolvimento de um sistema em PHP, para a gestão de quarteis de Bombeiros nos aeroportos de Cabo Verde.

O trabalho visa organizar todo o sistema de informação do serviço, concentra num único canal as informações necessárias para gestão. A organização e a digitalização dos Check-list das viaturas passaram a ser armazenadas em um banco de dados, reduzindo o tempo bem como o tratamento de informações. O sistema também irá ajudar e muito na organização dos extintores, uma vez que só no aeroporto de São Vicente dispõe de 120 extintores, que eram geridos em fichas de papel.

O protótipo esta dentro das necessidades atuais bem como os próximos 15 anos, podendo sofrer algumas atualizações, melhorando cada vez que se achar necessário.

Inicialmente o objetivo definido pelo trabalho era apenas um módulo com dois submódulos, pelo que durante o levantamento para início do desenvolvimento do sistema deparei que não era necessário o desenvolvimento do submódulo manutenção. Uma vez que o aeroporto já dispõe de um aplicativo para o fim, as alternativas foram aparecendo e ao terminar o levantamento deparei com a necessidade do desenvolvimento de pelo menos 10 submódulos para dar resposta a necessidade atual da gestão dos quarteis de Bombeiros.

Este trabalho foi de extrema importância para mim, visto que ao terminar a parte curricular do curso tinha muitas dificuldades em programação. Isto obrigou-me a dedicar por um período de cinco meses de estudos em PHP, Yii CSS, JQuery para que entendesse bem qual seria o caminho a percorrer.

Com o tempo dedicado aos estudos das linguagens mencionadas ajudou-me a aprofundar na linguagem PHP, entendendo-a de forma mais consistente e conhecendo uma nova linguagem de modelação, o JQuery.

Hoje graças as valências adquiridas durante o curso já posso pelo menos desenvolver uma aplicação mesmo que seja de pequena dimensão.

7.4 Proposta de melhoria

A aplicação apresentada é parte ou módulo de um sistema de gestão de quartel de Bombeiros, uma parte apenas do que será no futuro. Pode-se perfeitamente enquadrar este sistema em diversas empresas no mesmo modelo com pequenas alterações conforme a realidade introduzida.

Tendo em conta o local do uso desta aplicação irá necessitar de acompanhamento e atualização do sistema permanente, com intuito de melhoria contínua dos serviços e recursos disponibilizados, aquando necessário a introdução de novas funcionalidades.

Para este projeto as propostas futuras seriam a introdução de um módulo de gestão de emergência, em que seriam cadastrados as assistências prestadas pelo serviço. Todo o equipamento e consumíveis de emergência seriam introduzidos no sistema bem como dados disponíveis.

Também foi pensado em disponibilizar outro módulo de gestão de logística de equipamentos, ferramentas e lubrificantes. Após cadastrar dos itens será possível impressão de um relatório de itens do sistema e seu estado de conservação. A gestão dos lubrificantes, bem como o decremento da quantidade. O sistema emitirá um alerta quando qualquer dos lubrificantes estiver em falta ou por acabar.

7.5 Trabalhos Futuros

Como trabalho futuro deste sistema, pretendo que a tela de Dados seja dinâmico, seja ele na leitura dos combustíveis, os dados das assistências, para que seja uma plataforma estático de toda a informação que se achar necessário para o recetor.

O Check-List deverá se preenchido através de um dispositivo móvel, em que estará ligado diretamente aos servidores através de rede sem fios (ver figura 40).

O módulo de gestão em emergência devera ser preenchido numa ambulância através de um dispositivo móvel. Os dados serão armazenados no servidor sendo que o sistema estará ligado ao dispositivo móvel via internet.

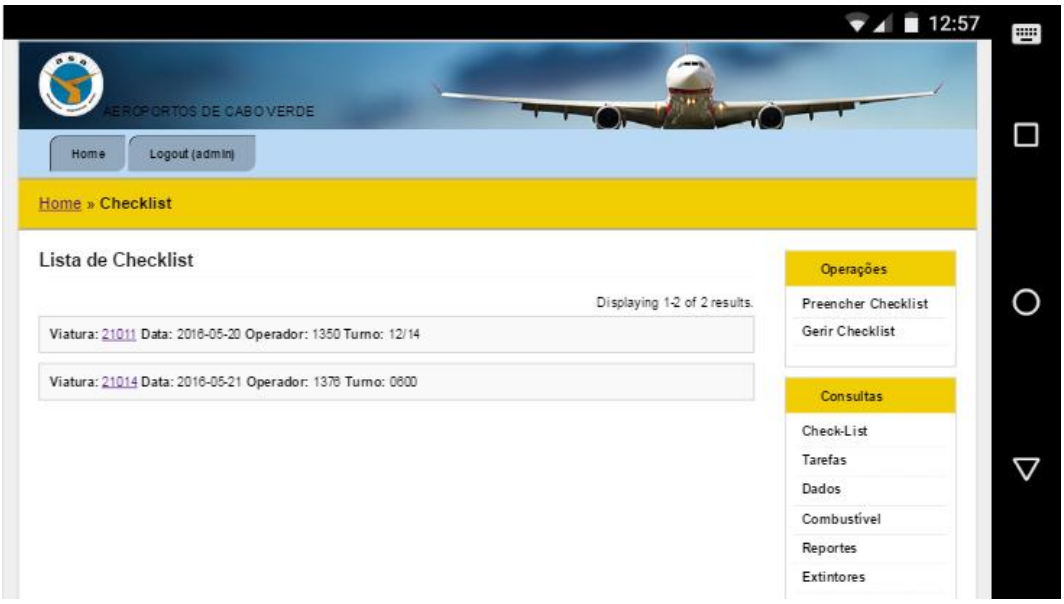


Figura 53 - Trabalho Futuro (O sistema a funcionar num sistema android)

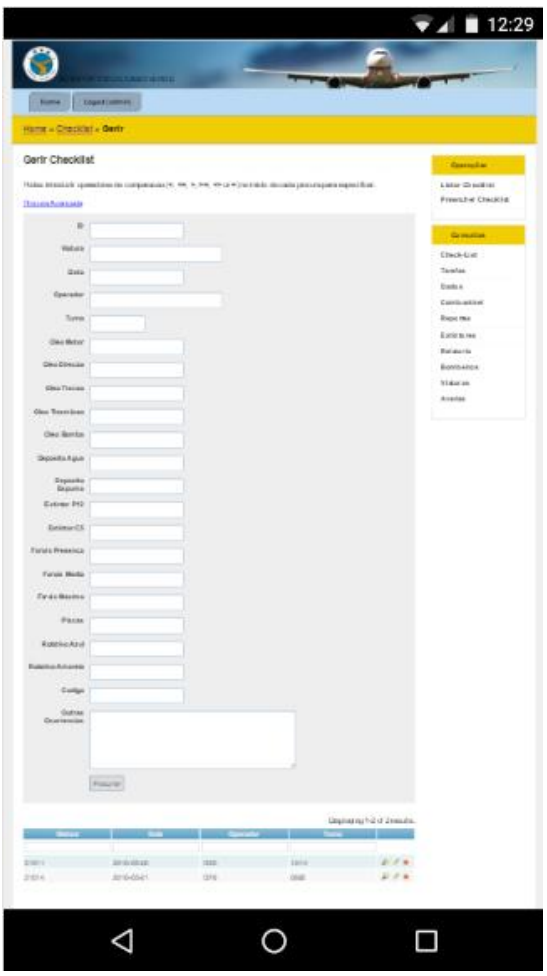


Figura 54 - Formulário de Pesquisa Avançada aberto num sistema android

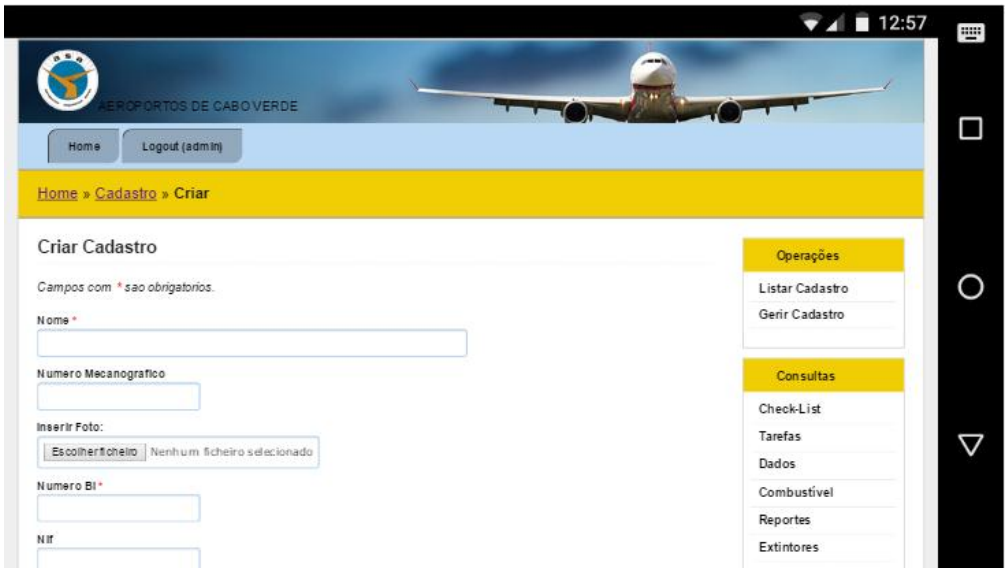


Figura 55 - Tela Cadastrar Bombeiro

7.6 Dificuldades Encontradas

Sendo que o Yii é um framework em constante evolução, ainda não é muito frequente encontrar documentos em português ou espanhol. Da minha parte inicialmente foi uma das barreiras encontradas, mas com o desenrolar do trabalho entendi que ao maior parte dos documentos referentes ao framework estava na página oficial mas em inglês. Recentemente há o tradutor automático da página o que facilita muito.

Para que este protótipo funcionasse, tive que dedicar alguns meses no estudo d PHP, JQuery, para entender como funcionava o gerador de código, desde a criação dos módulos e dos modelos. Também para entender o funcionamento do *view*, e a configuração de base do gerador, para que pudesse estar a vontade na manipulação dos códigos caso necessário.

7.7 Bibliografia

LOPES, F; M. Morais e A. Carvalho (2005). ***Desenvolvimento de Sistema de Informação***. FCA Editora de Informática.

MARQUES, Joaquim e SERRÃO, Carlos (2007). ***PHP5***. FCA Editora de Informática.

MIGUEL, António (2010). ***Gestão de projectos de software***. FCA – Editora de Informática.

PEREIRA, J. (1998). ***Tecnologia de Base de Dados***. FCA Editora de Informática.

SILVA, Alberto (2001). ***UML Metodologia e Ferramentas***. Editora Centro Atlântico.

7.8 Internet

1. (https://ifthensoftware.com/ProdutoX.aspx?ProdID=, 21-03-2016, 19:52)
2. (https://www.youtube.com/watch?v=U97pItIRHH0, 21-03-2016, 20:37)
3. (https://play.google.com/store/apps/details?id=com.appestry.firecast, 21-03-2016, 20:49)
4. (http://www.yiiframework.com/doc/guide/1.1/pt/quickstart.what-is-yii, 21-03-2016, 20:51)
5. (http://tableless.com.br/yii-framework-um-framework-php-profissional-rapido-e-seguro/, 21-03-2016, 20:56)
6. (http://www.devmedia.com.br/php-mvc-aplicando-o-padrao-mvc-no-php-yii-framework/31459, 21-03-2016, 20:59)
7. (http://www.yiiframework.com/doc/guide/1.1/es/quickstart.what-is-yii, 22-03-2016, 00:13)
8. (http://www.yiiframework.com/doc/guide/1.1/pt_br/changes/, 22-03-2016, 00:14)
9. (http://www.devmedia.com.br/php-yii-framework/30898, 22-03-2016, 00:28)

10. (http://www.yiiframework.com/doc/guide/1.1/pt_br/basics.model, 22-03-2016, 00:33)
11. (http://www.yiiframework.com/doc/guide/1.1/pt/basics.controller, 22-03-2016, 00:36)
12. (http://www.yiiframework.com/doc/guide/1.1/pt/basics.view, 22-03-2016, 00:40)
13. (http://tableless.com.br/entendendo-o-padrao-mvc-na-pratica/, 22-03-2016, 00:42)
    a. (https://www.webdevbr.com.br/mvc-e-php-entendendo-o-padrao-na-pratica-criando-um-framework-php, 22-03-2016, 00:49)
    b. (http://www.escolacriatividade.com/php-orientado-a-objetos-mvc-em-php/, 27-03-2016, 21:36)
14. (http://www.yiiframework.com/doc/guide/1.1/pt/basics.mvc, 22-03-2016, 21:50)
15. (http://www.yiiframework.com/doc/guide/1.1/pt/basics.component, 22-03-2016, 21:52)
16. (http://www.yiiframework.com/doc/guide/1.1/pt_br/basics.application#sec-4, 22-03-2016, 21:57)
17. (http://www.yiiframework.com/doc/guide/1.1/pt/basics.mvc, 22-03-2016, 22:08)
18. (http://www.yiiframework.com/doc/guide/1.1/pt_br/basics.application#sec-6, 22-03-2016, 22:11)
19. (http://conceito.de/sistema-de-informacao, 02-04-2016, 18:45)
20. (http://modelocascata.blogspot.com/, 02-04-2016, 18:57)
    a. (http://www.devmedia.com.br/introducao-ao-modelo-cascata/29843, s.d.)
    b. (http://pt.slideshare.net/erysonsi/modelo-cascata-apresentao, s.d.)
21. (https://pt.wikipedia.org/wiki/PHP, 02-04-2016, 19:01)
22. (http://www.significados.com.br/html/, 02-04-2016, 19:07)
23. (https://pt.wikipedia.org/wiki/JQuery, 07-04-2016, 22:19)

24. (http://www.tecmundo.com.br/programacao/2705-o-que-e-css-.htm, 07-04-2016, 22:21)
25. (https://www.oficinadanet.com.br/artigo/2227/mysql_-_o_que_e, 07-04-2016, 22:33)
26. (http://www.significados.com.br/sql/, 07-04-2016, 22:38)
27. (https://pt.wikipedia.org/wiki/Servidor_Apache, 07-04-2016, 22:41)
28. (http://www.infowester.com/servapach.php, 07-04-2016, 22:46)
29. (https://www.jetbrains.com/phpstorm/, 07-04-2016, 22:52)
30. (http://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408, 07-04-2016, 22:58)
31. (http://www.asa.cv, 07-04-2016, 23:15)
32. (http://www.unl.pt/guia/2012/fct/UNLGI_getUC?uc=7820, 07-04-2016, 23:29)
33. (https://pt.wikipedia.org/wiki/Diagrama_de_caso_de_uso, 07-04-2016, 23:44)
34. (http://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332, 05-05-2016, 20:14)
    a. (http://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332, 08-05-2016, 22:17)